IUT Paris Rives de Seine
BUT 2

J-M. Ilié , L. Giustignano, K. Foughali,
11 octobre 2023

TP AJAX

Communication HTTP en mode asynchrone

Dans ce TP, nous allons tester la possibilité de réaliser des requêtes ou demandes de services à des serveurs http distants. Les requêtes peuvent être paramétrées suivant les besoins. Il faudra exploiter les réponses de ces requêtes, notamment pour procéder à des affichages ou exploiter les données reçues, qui peuvent en différents formats, notamment textuelle ou JSON .

AJAX 1 : Un nombre aléatoire sur le net, voire une série de nombres :

Reprenez l'exercice du devin, de façon à tirer un nombre aléatoire grâce à cette API qui permet de tirer aléatoirement des nombres entiers.

https://www.random.org/clients/http/api/

Testez la requête suivante dans votre navigateur :

https://www.random.org/integers/?num=10&min=1&max=6&col=1&base=10&format=plain&rnd=new

Requete HTTP GET qui génère une série de num=10 entiers par défaut
pris entre min=1 et max=6
On peut préciser une base (base 10 par défaut)
et un nombre de colonnes (1 par défaut).
Valeur de retour en plain text ou html,
d'où nécessité de parser la valeur de retour (ParseInt pour un plain text).

AJAX 2 : Réception de données

Reprenez l'exercice du qcm, et notez que l'URL suivante renvoie le tableau `tabObject` contenant 3 qcms ou questions :

http://vs-wamp/dep_info/infoa2/qcm/qcm.php

Pour des questions de génie logiciel, tester cette URL dans la barre d'adresse du navigateur, préalablement à son utilisation dans une application cliente. Noter que la réponse du script PHP est alors l'émission d'une donnée JSON, contenant la description des 3 questions. Ce qui est reçu prend par défaut l'allure d'une chaîne de caractères. Pour afficher proprement de la donnée JSON reçue en affichant chacun de ses niveaux de façon hiérarchique, installer un plugin sur votre `navigateur`, comme l'extension `JSON Light` sur Firefox. Une alternative est l'application `postman`, qui permet de créer des requêtes http tout en gérant de façon distinguée chaque paramètre.

Voir par curiosité, l'écriture du script `qcm.php` écrit en `PHP` . Pour ce faire, l'adresse suivante est à saisir dans une fenêtre du système de gestion de fichiers du réseau du département informatique (explorateur window) :

```
\\vs-wamp\www\dep_info\qcm\qcm.php
```

Préparer maintenant une procédure dans votre application cliente, pour recevoir dans votre code la donnée `JSON` décrivant les questions, et affecter cette donnée à la variable `tabObject`, à l'initialisation de votre application ou tout du moins avant de commencer l'affichage de la question, par exemple dans le corps de la fonction `init` passé en paramètre de la fonction jquery `$.ready`:

```
function init() { // inclure l'appel à une procédure ajax }
$(document).ready(init).
```

Ecrire dans le corps de `init()` une requête AJAX opérant la réception de la donnée au format JSON par une transmission http de la requête en méthode GET, soit : .

```
$.getJSON("http://vs-wamp/dep_info/infoa2/qcm/qcm.php",
        function(dataJson){
            tabOject = dataJson;
            }
) ;
```

Du point de vue du fonctionnement, cette requête est par défaut réalisée en mode asynchrone par une thread dédiée, s'exécutant du côté du navigateur, avec une méthode de transmission `GET` du protocole `http` et telle que la donnée `JSON` reçue au terme de la requête est interprétée et allouée dans la mémoire du navigateur. Au terme d'un succès de la requête, la thread dédiée à son exécution invoque la fonction passée en paramètre, dite communément fonction callback. Elle rend directement exploitable la donnée reçue sous forme du paramètre de lafonction, noté ci-dessus `dataJSON`, d'où l'affectation directe à `tabObject`.

AJAX 3 : API de connexion surchargeant la description d'un formulaire HTML

Reprendre l'exercice du pavé numérique. Il s'agit de faire vérifier la saisie du formulaire par un script accessible via un serveur web répondant donc au requête HTTP. Pour la suite, considérer que le formulaire est composé de 2 champs HTML <input> caractérisés respectivement par les attributs `name= 'nom'` et name= 'code', soit :

```
<form id="f1" action="">
      <input name="nom"/>
      <input name="code"/>
</form>
```

Déclarer que sur un événement submit l'exécution d'une fonction *callback* prenant en

charge la procédure Ajax sera exécutée, conduisant donc à la soumission du formulaire. Ne pas oublier d'empêcher le navigateur de poursuivre la soumission du formulaire par lui-même. Pour ce faire, terminer cette fonction par un "`return false`" ou en utilisant la fonction `preventDefault` attachée à l'événement de soumission, ci-dessous noté `ev`:

```
$("#f1").on( "submit", function(ev) {
            ev.preventDefault();
           //A faire ensuite : la requête Ajax
           //de soumission du formulaire
});
```

Sur le serveur web du département, le script interprétant les données saisies est accessible via une requête http, suivant la méthode de transmission `post`, à l'url :

http://vs-wamp/dep_info/infoa2/clavierNum/connexionAjax.php

Par curiosité, voir l'écriture de ce script PHP à partir d'une fenêtre de l'explorateur window, à l'adresse suivante :

```
\\vs-wamp\www\dep_info\infoa2\clavierNum\connexionAjax.php
```

Utiliser dans votre application cliente une procédure ajax réalisant une requête http vers cette url en mode asynchrone, par exemple `$.ajax()` de `jquery` . Préciser ses paramètres sous la forme d'un unique objet au format JSON :

```
$.ajax ({

      url : "http://vs-wamp/dep_info/infoa2/clavierNum/connexionAjax.php",
      method : "POST",
      data : paramIN, //à préciser
      datatype : "html", // format retour: html (text), json…
      success: function(d) { //exploiter d; },
      error :     function(e) {  //exploiter e}

      }) ;
```

- `paramIn` est ici une variable qui présente l'ensemble des champs saisis du formulaire, sous la forme d'une chaîne de caractères au format des paramètres d'une url. Noter que la fonction `$.serialize()` de jquery rend automatiquement cette chaîne, quelle que soit les valeurs saisies. En conséquence, exploiter l'une des 2 écritures suivantes:

```
let paramIn = "nom=bond&code=000007"; //en dure !! ou …

let paramIn = $("#f1").serialize() ;
```

- Préciser le corps de la fonctions callback représentée pour `success`. Le paramètre `d` utilisé correspond pour l'exécution de cette fonction à la réponse du script PHP. Il aura comme valeur `'OK'` ou `'KO'`, ce qui vous permettra d'afficher soit une "bienvenue" à l'utilisateur Bond soit une invitation à recommencer la saisie de la connexion si le nombre de soumission possibles n'est pas atteint (3 par exemple).
- Préciser le corps de la fonctions callback représentée pour `error`, pour prévenir l'utilisateur d'une erreur inattendue lors de l'exécution du protocole.
- Note : une alternative à `$.ajax()` serait d'utiliser la méthode `fetch()` de Javascript.

AJAX 4 : quel temps fait-il quelque part ?

Le site web `openweather` propose de vous informer de la météo à partir des coordonnées d'une position géospatiale du globe. Une API est accessible sous la condition d'une inscription au site.

Informer vous sur cette API, ses paramètres et s'inscrire pour récupérer la clé à préciser dans dans toutes vos requêtes par un paramètre nommé `appId` .

https:/ /openweathermap.org/current

- Développer une application cliente qui exploite ce service. Proposer à l'utilisateur de fournir les coordonnées du lieu.
- Améliorer votre application en proposant à l'utilisateur de fournir une adresse plutôt que des coordonnées. Le service gratuit `nominatim.org` du serveur coopéartif `openstreetmap` fournit 2 API `/search` et `/reverse`, qui permettent respectivement de fournir les coordonnées géospatiales d'une adresse et réciproquement de fournir l'adresse à partir de coordonnées géo-spatiales. Lire la documentation des API de nominatim :

  https://nominatim.org/

A titre d'essai, tester l'url suivante sur votre navigateur permettant notamment d'obtenir les coordonnées géospatiales d'un point central de la France, et celles d'un rectangle englobant noté boundingbox:

http://nominatim.openstreetmap.org/search?country=france&limit=1&format=json&addressdetails=1

Résultat retourné au client au format JSON :

[{"place_id":"9157173744",
"licence":"Data \u00a9 OpenStreetMap contributors, ODbL 1.0.http:\/\/www.openstreetmap.org\/copyright",
"osm_type":"relation","osm_id":"2202162",
"boundingbox" : ["-85.0500030517578","51.2683181762695","-178.38737487793","172.305725097656"],
"lat":"46.603354",
"lon":"1.8883335",
"display_name" : "France"}}]

- Visualiser graphiquement les positions retournées en utilisant par exemple `google map`, en donnant à rechercher une expression réduite aux notions de latitude et longitude :

```
46.603354  1.8883335
```

Développement à faire pourune application cliente :

- Prévoir la saisie d'une adresse et d'un bouton pour demander la météo pour cette adresse (la validité de l'adresse pourra être vérifiée en aparté sur google ou osm).

L'écouteur de clic sur le bouton lançera une fonction `meteo` à écrire, faisant la recherche meteo à partir de l'adresse valide saisie.

- La fonction meteo aura comme 1er objectif de lancer une procédure Ajax pour récupérer les coordonnées géospatiales de l'adresse saisie, plus exactement dans le corps de la fonction notée `success`, il s'agira d'exploiter la donnée json reçue en paramètre et d'en extraire les valeurs `lat` et `lon`.
- Le second objectif de la fonction meteo sera d'invoquer le service météo paramétré de `openWeather` afin d'afficher à l'utilisateur les informations météorologiques pour les coordonnées désirées.