TP Labyrinthe : partie 2

Rédigé par : Jimmy Paquereau

1. Préliminaire

Au dernier épisode, nous avons entrevu quelques-unes des technologies du web : le HTML, le CSS et le JavaScript. Nous avons encore quelques principes autour de la notion d'architecture client-serveur. Cependant, il importe de bien avoir compris que les modestes lignes que nous avons écrites en HTML, CSS et JavaScript, sont interprétées par votre navigateur. De ce fait, nous n'avions pas nécessairement besoin d'un serveur web.

Désormais, nous allons nous intéresser plus amplement aux interactions client-serveur. Le **navigateur** (Chrome) demeurera le client. Le logiciel que vous avez normalement installé (**Wampserver 2.5**) fera office de serveur web. Quant aux algorithmes, nous développerons sous **Notepad++**. De fait, WAMP est un acronyme pour :
o **Web** ;
o **Apache** : le serveur web à proprement parler ;
o **MySQL** : le SGBD sur lequel placer une ou plusieurs base de données ;
o **PHP** : le langage de programmation utilisé côté serveur.

A la fin de la séance, vous êtes supposés avoir compris et/ou manipulés les notions suivantes :
o quelques éléments du protocole HTTP ;
o mise en œuvre de requêtes SQL simples au sein d'algorithmes ;
o mise en œuvre d'algorithmes simples en PHP ;
o principe et intérêt d'une architecture client-serveur ;
o etc.

N.B. : on notera que la deuxième partie du TP n'a visuellement aucun rapport avec le jeu du labyrinthe qui vous a été présenté. Cependant, celui-ci possède bien une interface d'inscription, des joueurs… Bref, nous nous contenterons de quelque chose de bien plus simple.

2. Travail à réaliser

Afin d'effectuer le travail du jour, vous aurez à votre disposition quelques ressources contenu dans une archive ZIP (**tpWeb.zip**).

1. Création d'un virtualhost

Notre serveur web en place, nous allons créer ce que l'on appelle un **virtualhost**. C'est un service virtuel qui va vous permettre d'accéder à votre serveur web local via un **nom de domaine*** virtuel (à savoir un nom de domaine accessible uniquement via votre ordinateur) et un **port***. Un nom de domaine est un équivalent d'une adresse IP. Les serveurs DNS (*Domain Name System*) assure la **translation d'adresse** (nom de domaine vers IP et IP vers nom de domaine).

* On rappelle que le couple **(nom de domaine, port)** permet d'identifier un service sur le réseau, le nom de domaine permettant d'identifier l'hôte du réseau, le port permettant d'identifier le service. Ici, le réseau, c'est le même principe mais le réseau se cantonne à votre propre ordinateur, d'où l'a

Etapes : effectuées ensembles

o création de l'hôte virtuel dans le fichier %WAMP%/bin/appache/conf/extra/httpd-vhost.conf :
Nom de domaine : **tpWeb.local** Port : **8080** Dossier cible : **tpWeb/**

o modification du fichier c:/windows/system32/drivers/etc/host, ajout des lignes suivantes :
 # tpWeb
127.0.0.3 tpWeb.local

o (re)démarrage de WampServer.

Rappel : les adresses IP en 127.X.X.X sont des **adresses de loopback**. Dès que vous envoyez une requête (http ou autre) vers cette adresse, la requête est redirigée vers votre ordinateur.

Dans votre navigateur favori, saisissez dans la barre d'adresse : http://127.0.0.3/, puis appuyez sur la touche entrée.

Question 1.1 : quel résultat obtenez-vous ? Pouvait-on s'attendre à un autre résultat ? Justifier ce résultat.
Question 1.2 : que faut-il corriger pour espérer pouvoir obtenir un résultat ? Justifier votre choix.
Question 1.3 : peut-on obtenir un résultat sans se servir de l'adresse IP 127.0.0.3 ? Si oui, proposer une solution et justifiez votre choix (comment cela fonctionne-t-il ?).

2. Méthodes et paramètres HTTP

Question 2.1 : rappelez quelles sont les principales méthodes HTTP.

Le fichier **httpinfo.php** est accessible via l'URI /httpinfo.php, à savoir via l'URL http://tpweb.local/httpinfo.php. Une URI n'est en fait qu'une URL partielle, relative à un nom de domaine. Dans la suite du TP, on emploiera plus volontiers les URI (car plus courtes).

Saisir dans la barre d'adresse URI : /httpinfo.php?prenom=joseph&nom=fourier

Question 2.2 : quelles sont les informations contenus dans cette URI ?
Question 2.3 : quelles informations transparaissent dans le résultat affiché ?

Effectuez à présent la manipulation suivante :
o actualisez la page de votre navigateur ;
o sur votre page, effectuez un clic droit puis « inspecter l'élement » ;
o allez dans l'onglet « network » ;
o dans cet onglet, repérez la ligne correspondant à votre requête HTTP (httpinfo.php) ;
o effectuez un clic droit, « copy request headers » puis collez dans une nouvelle page notepad++ ;
o faite de même mais cette fois-ci en choisissant successivement « copy response headers » et « copy response ».

Question 2.4 : les informations que vous avez copiées dans notepad++ sont les messages (requête et réponse HTTP) que se sont transmises le client et le serveur. En vous appuyant en outre sur le cours, expliquez-en le contenu.
Question 2.5 : à l'écran, le résultat de votre précédente requête HTTP est mis en forme. Qu'est-ce qui justifierait la présence d'une pareille mise en forme ?
Question 2.6 : en vous servant de la manipulation précédente, justifier clairement votre réponse à la question 2.5.
Question 2.7 : éditez le fichier **httpinfo.php** avec NotePad++ (clic droit > edit with notepad++), expliquez-en l'algorithme (expliquez en outre la signification de chacun des éléments du fichier : variables, boucle, balise…).

Le fichier **subscription.php** est quant à lui accessible via l'URI /subscription. Testez le formulaire (complétez puis validez le formulaire) puis observez le contenu des fichiers **subscription.php** et **subscribe.php**.

Question 2.8 : expliquez brièvement ce qui s'est passé (le cheminement général).
Question 2.9 : quelle méthode HTTP est utilisée ? Quels types de paramètres sont utilisés ?
Question 2.10 : à votre avis, quel est l'inconvénient des paramètres GET ?

3. Serveur web et base de données

Tout d'abord, il nous faut une basse de données. Aussi, nous allons créer la base de données et y insérer quelques jeux de test :
o dans votre *system tray* (icônes en accès rapide), faites un clic gauche sur l'icône de WampServer (icône « W ») ;
o cliquez sur l'item phpMyAdmin. La fenêtre de PhpMyAdmin s'ouvre dans le navigateur ;
o créez une nouvelle base de données, appelé **tpweb** ;
o sélectionnez la base de données, puis allez dans l'onglet SQL ;
o glissez-déposez le fichier **tpweb.sql** dans l'éditeur, puis cliquez sur « exécuter » ;
o actualisez la page.

Informations complémentaires :
o la durée d'une partie est exprimées en millisecondes ;
o le champ « level » de la table « Party » prend les valeurs suivantes : « EASY », « MEDIUM » et « DIFFICULT ».

Question 3.1 : en cliquant sur les tables et en vous servant de l'onglet structure (et du lien « vue relationnelle » de cet onglet), déterminez le MCD et le schéma relationnel de cette mini base de données.
Question 3.2 : dans l'éditeur SQL, rédigez les requêtes permettant :
o d'afficher les parties du joueur « Joseph FOURIER » ;
o d'afficher la liste des parties en cours où le joueur 1 est en attente d'adversaire ;
o d'afficher le nombre de parties non commencées ;
o d'afficher la durée moyenne d'une partie (en secondes) ;
o d'afficher le nombre de parties jouées par « Joseph FOURIER » ;
o d'afficher le nombre de parties remportées par « Joseph FOURIER ».

Le fichier **players.php** est accessible via l'URI /utilisateurs. Son algorithme permet d'afficher une liste d'utilisateurs. Testez l'URI.

Question 3.3 : expliquez à présent le contenu du fichier **players.php**.
Question 3.4 : en particulier, expliquez ce qui va se passe au clic sur l'icône « voir » sur chacune des lignes utilisateur.

Question 3.4 : à l'aide des questions 3.2 et 3.3, complétez le fichier **player.php**, lequel permet d'affichez le détail des informations relatives à un utilisateur. On souhaite pouvoir consulter les informations suivantes :
o le détail du joueur (déjà fait) ;
o la liste des parties effectuées par celui-ci ;
o son nombre de victoires et de défaites.
o toute information qui vous semble à propos ;
o toute mise en forme qui vous semble à propre (C'est secondaire !).

4. Evolutions

Quoique, en pratique, on se stockerait très probablement pas de telles informations en base de données, on souhaiterait désormais stocker :

o le chemin parcouru par chacun des joueurs au cours d'une partie (évolution 1) ;
o des informations suffisantes afin de pouvoir reproduire le labyrinthe sur lequel les joueurs se sont défiés (évolution 2).

Question 4.1 : compléter le MCD précédent de manière à prendre en charge l'évolution 1 (contrainte : deux solutions devront être trouvées : une solution non récursive et une solution récursive).
Question 4.2 : compléter le MCD de sorte que l'évolution 2 soit prise en charge.

Rappels :
o **NOW**() : retourne la date courante ;
o **YEAR**(champ), **MONTH**(champ), **DAY**(champ) : retourne respectivement l'année, le mois et le jour correspondant à la date champ ;
o **HOUR**(champ), **MINUTE**(champ), **SECOND**(champ) : retourne respectivement l'heure, la minute et la seconde correspondant à l'heure champ.

Question 4.3 : des dates et toujours des dates. Rédiger les requêtes permettant :
o de passer un utilisateur à l'état « déconnecté » s'il est connecté depuis plus d'un jour ;
o d'afficher les utilisateurs qui ne se sont pas connectés depuis plus d'un an ;
o à l'aide d'au moins une sous-requête, de supprimer les utilisateurs qui ne se sont pas connectés depuis plus d'un an (soit 365 jours, qu'importe les années bissextiles).

Question 4.4 : une histoire de chemins. Rédiger les requêtes permettant :
o d'afficher le ou les utilisateurs ayant terminé la partie le plus rapidement en mode « difficile » ;
o d'afficher le record pour chaque mode ;
o d'afficher les recordmans pour chaque mode.

Conclusion : bon courage ! Et si vous parvenez à comprendre et à réaliser seul toutes les questions de ce TP, c'est que le P10 ne devrait vous poser aucun problème au BTS.