Programmation web avancée

Yacine Bouzidi

May 29, 2017

Familiarisation avec la bibliothèque JQuery

À la fin de cette séance, chaque étudiant devra déposer ses fichiers sur le dépôt moodle.

1 Exercice 1

- Définir une classe "allume";
- Définir plusieurs balises (p, div...) de la classe "exemple" cachés ;
- Écrire une fonction "decollage", appelée lors du clic sur un bouton
  - Son contenu : $("p.exemple").addClass("allume").show("slow");

2 Exercice 2

- Modifier le label et le comportement du bouton pour qu'il permette de cacher les éléments qui avaient été affichés ;
  - Créer plusieurs boutons synchronisés entre eux pour cacher/afficher les éléments (un en haut et l'autre en bas)
  - Insérer des boutons permettant d'ajouter de nouveaux paragraphes à la page, placés avant ou après les paragraphes existants

3 Exercice 3

- Des boutons '+' seront affichés par JQuery de part et d'autre d'un div "contenu"
  - Ils permettront de créer de nouveaux pararaphes lors d'un click
- Les paragraphes créés seront éditables
  - Pour être éditable, un paragraphe aura juste besoin d'être de la classe "éditable"
- **Version 1** : Si on clique sur un autre paragraphe alors que le premier est en cours d'édition, rien ne se passe

  - Si on appuie sur la touche Entrée, on valide la saisie
  - Si on appuie sur la touche ESC, on annule l'édition en cours en réaffichant l'ancien contenu

- **Version 2** : Plusieurs paragraphes sont éditables en même temps
  - On utilisera $().DATA pour stocker les valeurs des anciens contenus à restaurer lors de l'appui sur 'ESC'

4 Exercice 4 : Utilisation d'une base de donnée

- Récupérer des paragraphes depuis une base de données
  - Les rendre éditables
- A chaque modification, envoyer la valeur du paragraphe édité au serveur pour le sauvegarder en base de données
  - Interdire l'envoi d'un paragraphe vide
- Ajouter des fonctionnalités :
  - Création de nouveaux paragraphes
  - Suppression d'un paragraphe

5 Exercice 5 : Plateforme de micro-blogging

Cet exercice consiste à mettre au point un système de microblogage de type Twitter avec php (ou autre), javascript et ajax. En quelque mots, ce système permet aux utilisateurs inscrits d'envoyer des messages instantanés (tweets) mais également de visualiser des messages envoyés par d'autres utilisateurs. Outre l'exigence de concision, une différence majeure entre Twitter et les systemes de blogage traditionnels réside dans son aspect réseau social, c.a.d permettant de connecter différents utilisateurs entre eux. Ce lien que l'on appellera ici *abonnement* permet à chaque utilisateur de recevoir sur sa page les messages postés par des utilisateurs auxquels il est abonné. Ainsi, si l'utilisateur Alice est abonné à l'utilisateur Bob, tout message posté par Bob apparaitra automatiquement dans la page d'Alice. En pratique, Twitter dispose de nombreuses autres fonctionnalités; cependant pour des raisons de temps, il ne sera pas question de toutes les reproduire ici. Aussi l'objectif principal de ce tp sera d'en implémenter une version simple modélisant l'envoi d'un message par un utilisateur et sa visualisation par les différents abonnés.

5.1 Préliminaires

Base de donnée. Chaque étudiant devra mettre en place une base de donnée Mysql hébergée sur son serveur local. Cette base de donnée contient les tables suivantes :

TWITTER_USERS	
Attribut	Type
id	int(10) unsigned
login	varchar(255)
mdp	varchar(255)
nom	varchar(255)
prenom	varchar(255)
mail	varchar(255)

TWITTER_TWEETS	
Attribut	Type
idT	int(11)
idU	int(11)
tweet	text
horaire	datetime

TWITTER_FOLLOWERS	
Attribut	Type
idU	unsigned
idF	unsigned

Arborescence des fichiers. Les fichiers du projet doivent êtres stockés dans un répertoire appelé Tp_Twitter, à l'intérieur duquel plusieurs autres répertoires doivent êtres crées :

- css,
- datas (contiendra les données du site, comme mysql.php, les images, les parties de code en html à conserver dans des fichiers séparés),
- essais (contiendra tous les scripts que vous mettrez au point, ceci afin de ne pas polluer les fichiers déjà validés),
- js (contiendra tous les fichiers javascript dont vous aurez besoin),
- reponsesAjax (contiendra tous les fichiers php appelés au moyen des techniques ajax),
- utils (contiendra tous les fichiers php contenant des fonctions que vous utiliserez).

5.2 Implantation

5.2.1 Authentification

Il s'agit pour commencer de mettre en place un script de connexion à la session. Cette connexion se fait au moyen d'un formulaire de saisie de login et de mot de passe qui interroge la base de donnée pour vérifier que l'utilisateur est bien enregistré. Le but de cet exercice est d'arriver à rendre l'utilisation de l'authentification systématique. Chaque page du site, devra en conséquence contenir un script qui vérifie que l'utilisateur est bien authentifié et le cas contraire, le renvoyer vers la page d'authentification. Lors de la connexion, Vous devez mettre à jour la variable globale $_SESSION pour conserver l'identifiant de l'utilisateur, son nom, son prénom ainsi que l'heure de sa dernière connexion. De plus, à chaque exécution du script d'authentification vous devrez également vérifier si la session est ouverte depuis plus de 30 mins. Le cas échéant, cette dernière doit être détruite, sinon l'heure de connexion est mise à jour à l'heure courante, pour de nouveau laisser 30 min à l'utilisateur.

5.2.2 Envoi d'un message

Après authentification, un utilisateur est envoyé sur sa page d'accueil. Celle-ci contient un formulaire de saisie de messages (tweets) ainsi qu'une zone ("div") permettant d'afficher les messages envoyés par d'autres utilisateurs. Dans un premier temps nous allons nous concentrer sur l'envoi d'un message par un utilisateur, nous aborderons l'affichage des messages un peu plus tard dans ce tp. Ajoutez dans le répertoire js une

fonction ajax permettant d'ajouter à la table Twitter_tweets un message saisi par un utilisateur. Le traitement de la requête se fera à l'aide d'un fichier PHP qui sera stocké dans le répertoire reponse_ajax.

5.2.3 Abonnement

La notion d'abonnement est décrite dans ce tp par la table TWITTER_FOLLOWERS, cette table contient des couples d'identifiants d'utilisateurs (idU,idF) qui relient un utilisateur aux autres utilisateurs abonnés à ses messages i.e. l'existence d'un couple (id1,id2) dans la table TWITTER_FOLLOWERS signifie que l'utilisateur id2 est abonné aux messages de l'utilisateur "id1". Pour permettre à un utilisateur de s'abonner/se désabonner au messages d'un autre utilisateur, nous proposons de rajouter les trois fonctions suivante :

- Une fonction **Recherche**, permettant de retrouver un utilisateur à l'aide de son login, son nom et son prénom. Pour cela vous devez mettre en place un formulaire de recherche accessible depuis la page d'accueil, une fonction ajax qui permet d'envoyer les données saisies (login, nom et prénom) et un script PHP permettant le traitement de la rêquete et le renvois du résultat.

- Une fonction **Abonnement** permettant à partir du résultat de la recherche de s'abonner au messages d'un utilisateur. Pour cela, un bouton "s'abonner" devra apparaître devant l'utilisateur retourné par la recherche, ce dernier lance l'exécution d'une fonction ajax qui rajoute le lien d'abonnement entre les deux utilisateurs.

- Enfin, une fonction **Désabonnement** qui comme son nom l'indique permet à un utilisateur de se désabonner des messages d'un autre utilisateur. Celle-ci aussi devra être développée selon le principe d'ajax.

5.2.4 Affichage des messages

Nous allons à présent aborder le problème de l'affichage des messages. Selon le principe de Twitter, un utilisateur doit voir afficher sur sa page d'accueil ses messages ainsi que ceux des utilisateur auxquels il est abonné. Pour affichez ces messages, vous devrez rajouter dans le fichier correspondant à la page d'accueil une fonction ajax qui fait appel à un script PHP. Ce dernier, au moyen d'une requête sql sélectionne les messages qui doivent être affichés puis le renvois sous la forme d'une liste. Le format de cette liste n'est pas contraint, cependant il est recommandé de renvoyer du code html à intégrer directement dans la "div" réservée à cet effet. Par ailleurs, l'affichage des messages est effectué de maniere automatique et à intervalles de temps réguliers, par conséquent la requête ajax doit elle aussi être exécuter de manière automatique. Pour cela vous utiliserez la fonction javascript SETINTERVAL() :

```
id=setInterval("viewTweets()", 10000);)
```

Cette fonction prend deux arguments. Le premier est une instruction javascript donnée sous forme d'une chaine de caractères (ici l'appel de la fonction VIEWTWEETS()); et le deuxième argument est le délai en millisecondes entre chaque exécution de l'instruction. Le fait d'affecter le retour de cette fonction à une variable (ici la variable id) vous permettra d'arrêter cette exécution périodique au moyen de la fonction CLEARTIMEOUT(ID).

5.2.5 Répondre aux messages

Pour finir, on souhaite offrir la possibilité aux utilisateurs de répondre aux messages d'autres utilisateurs auxquels ils sont abonnés. Pour cela, à l'affichage des messages, on rajoutera une zone de texte en dessous de chaque message pour saisir la réponse à celui-ci. Modifiez votre base de donnée en ajoutant éventuellement de nouvelles tables pour prendre en compte le lien "réponse" entre les messages et ajoutez une fonction ajax pour valider les réponses au message. Il est à noter enfin qu'à l'affichage d'une page utilisateur, les réponses aux messages de celui-ci seront par défaut cachées. Ces dernières pourront êtres rendues visibles en cliquant sur une bouton "voir réponses".