Sujet 10 - Technologies du Web

avril 2020

Pages dynamiques et login/logout

Cet exercice est la prolongation de ceux des 2 fiches précédentes. Vous aurez besoin

- dans la base de données : les tables de la course cycliste ET la table `s8_users` (fiche 8) avec des mots de passe cryptés.
- de l'arborescence de fichiers ci-joint. Elle complète les fichiers qui vous étaient donnés pour la fiche 9
  - **conservez les fichiers que vous aviez écrits, installez la nouvelle arborescence dans un autre dossier**.
  - compétez avec votre fichier `lib/db_parms.php`
  - mettez à jour `lib/session_start.php` avec le nom que vous utilisez pour la session.
  - examinez le contenu des fichiers qui vous sont donnés. Vous y trouverez de nombreux éléments de solution des fiches précédentes.

Les services à écrire devront produire une réponse ayant la même structure que celle décrite dans la fiche 9.

Exercice 1 :
Q 1 .

1. Complétez le script `services/login.php` qui permet à un utilisateur de se connecter.
   Il attend les arguments obligatoires `login` et `password` et permet d'ouvrir une session authentifiée, comme vous l'aviez fait pour la fiche 8. Il doit donc
   - si `$_SESSION['ident']` était déjà positionné, le service doit produire une erreur ;
   - sinon, si le login et le password fournis sont incorrects, il produit également une erreur
   - si le login et le password fournis sont corrects, il positionne `$_SESSION['ident']` avec une instance de `Identite`. L'attribut `result` de la réponse du service est un objet contenant les informations sur la personne connectée (`login, nom, prenom`).

   NB : provisoirement, pour faciliter les tests, le service acceptera les arguments envoyés en mode `get` ou `post`. Dans la version définitive il ne prendra que les arguments envoyés en `post`.
2. Le script `services/logout.php` vous est fourni ; Il n'attend aucun argument. Il met fin à la session en cours. Si aucune session n'était en cours, il produit une erreur.

Testez les scripts en entrant directement leur URL et arguments dans la barre d'URL du navigateur. Vérifiez notamment qu'un login suivant immédiatement un autre, sans logout entre les deux, produit une erreur.
Une fois les services testés et validés, supprimez la prise en compte du mode GET pour le login.

Q 2 . Le script `complet.php` est une version complétée de `multi.php` (feuille précédente).
Il fait appel à la page `views/pageComplet.php`. On y trouve :

- des éléments qui ne devraient être affichés que quand l'utilisateur est « connecté » (c'est à dire authentifié). Ces éléments sont marqués sémantiquement par la classe (au sens HTML) `connecte`.
- des éléments qui ne devraient être affichés que quand l'utilisateur est « déconnecté », marqués sémantiquement par la classe `deconnecte`. Cette partie comporte notamment un formulaire de login.

Cette même page peut donc prendre deux aspects un peu différents selon la situation. Des fonctions javascript permettent de basculer entre les deux états possibles.
Pour cacher un élément, on utilise l'attribut `hidden`.

1. dans le navigateur, assurez-vous d'abord que vous que vous n'êtes pas dans une session authentifiée. Au besoin, utilisez `services/logout.php` pour vous déconnecter.
   Ensuite chargez la page `complet.php`.

2. prenez connaissance du fichier `js/gestion_log.js`
   Les fonctions `etatConnecte()` et `etatDeconnecte()` permettent de basculer entre les 2 états. Initialement, la page est mise dans l'état « non connecté ».
3. écrivez la fonction `sendLogin()` qui envoie (par fetch) les données du formulaire au service `services/login.php`, en mode POST. Faites en sorte que si l'authentification est réussie, la page bascule dans l'état connecté.
   NB : pour toutes les requêtes fetch utilisant les sessions (donc les cookies), ce qui est le cas ici, il faut utiliser l'option `credentials:'same-origin'`.
4. écrivez la fonction `sendLogout()` qui envoie par fetch une requête à `services/logout.php`. Faites en sorte que si la déconnexion est réussie, la page bascule dans l'état déconnecté.

Testez le fonctionnement du login et du logout.
Q 3 . Il reste un petit problème, comme vous allez le constater : **connectez-vous**, puis rechargez la page. La page rechargée est alors dans l'état déconnecté, alors qu'en réalité vous êtes toujours connecté. Si vous essayez de vous connecter, cela ne fonctionne d'ailleurs pas (en principe ; ou alors ce n'est pas normal).

1. modifiez `views/pageComplet.php` pour dé-commenter : `if(isset($personne)) $dataPersonne=...`
   Quand l'utilisateur est authentifié, la balise `<body>` comportera maintenant un attribut `data-personne` qui contient l'objet Personne (en JSON, bien-entendu). (vous pouvez recharger une nouvelle fois la page et constater avec l'inspecteur Firefox que c'est bien le cas).
2. modifiez la fonction javascript `initState()` pour tenir compte de l'existence (ou pas) de cet attribut : s'il est défini, la page doit être passée dans l'état « connecté », en utilisant l'objet personne présent dans l'attribut ; dans le cas contraire, la page doit être mise dans l'état « non connecté ».

   Rappels :
   - en js, la valeur d'un attribut HTML `data-xxx` est contenue dans `dataset.xxx` (vous avez probalement utilisé cela pour le projet 1). Donc ici : `document.body.dataset.personne`
   - pour transformer un texte JSON en objet : fonction `JSON.parse()`

Rechargez une nouvelle fois la page pour tester le fonctionnement (testez des connexions et déconnexions en intercalant des rechargements de page).

Exercice 2 : *Supplément facultatif*
Vous ajouterez à la base de données une nouvelle table `favoris` dont la commande de création est dans le fichier `creationFavoris.sql`
Q 1 . Chaque utilisateur peut choisir un ou plusieurs coureurs favoris. Ces choix figurent dans la table `favoris`.
Avec PhpPgAdmin, créez « manuellement » plusieurs enregistrements dans cette table.

1. créez `services/getMesFavoris.php` : pour un utilisateur déjà authentifié, ce service fournit la liste de ses coureurs favoris (aucun argument n'est donc nécessaire). Si la requête n'est pas faite dans le cadre d'une session authentifiée, le service produit une erreur.
   Vous utiliserez la méthode `getFavoris()` de la couche données (consultez sa spécification).
2. complétez le script js pour faire en sorte que lors du passage dans l'état connecté, on affiche la liste des coureurs favoris, dans l'élément `liste_favoris`

Q 2 . Dans `views/pageComplet.php` décommentez les formulaires permettant à l'utilisateur d'ajouter ou de supprimer des coureurs favoris, puis complétez le javascript pour les faire fonctionner. La liste de coureurs proposée par chaque formulaire sera adaptée à la situation (ne pas proposer d'ajouter des coureurs qui sont déjà favoris ni de supprimer des coureurs qui ne le sont pas). À chaque ajout ou suppresion, il faut mettre à jour la liste affichée ainsi que les listes de choix des formulaires.
Vous pouvez ajouter au service `getMesFavoris.php` un argument optionnel `opposite` valant `true` ou `false` (défaut). Si `opposite` vaut true le service fournit la liste des coureurs **non** favoris.