

TP9 – Connexion au site – rôle d'administrateur

Ce que vous voyez du site après le TP8, c'est plutôt une vue « admin ». En effet, c'est plutôt un privilège d'administrateur que de créer, modifier, supprimer les adhérents, les auteurs, les livres, etc.



On peut imaginer, si un adhérent n'est pas administrateur, qu'il puisse accéder à son profil, le modifier, mais pour les livres, ne consulter que la liste (et plus tard, faire des demandes d'emprunt, mais ce n'est pas encore le moment).

On peut aussi imaginer, si l'adhérent n'est pas connecté, qu'il n'ait accès qu'à l'interface de connexion ou de création de compte. Il va donc falloir créer une « surcouche » à notre site, qui nous permettra de gérer les **sessions**.

Pour concrétiser la différence administrateur / non administrateur, on va considérer que l'un des adhérents est un adhérent spécial, c'est l'administrateur. Pour concrétiser cela, on va modifier la table « Adherent » dans la base de données en introduisant un nouveau champ « isAdmin ».

Exercice 1 – adhérent administrateur ou simple adhérent ?

1. Par l'interface phpMyAdmin, modifiez la table Adherent avec la structure suivante :

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut
login 	varchar(50)	utf8_general_ci		Non	Aucun(e)
mdp	varchar(64)	utf8_general_ci		Non	Aucun(e)
nomAdherent	varchar(50)	utf8mb4_general_ci		Oui	NULL
prenomAdherent	varchar(50)	utf8mb4_general_ci		Oui	NULL
email	varchar(50)	utf8mb4_general_ci		Oui	NULL
dateAdhesion	date			Oui	NULL
numCategorie 	int(11)			Non	Aucun(e)
isAdmin	tinyint(4)			Non	0

2. Modifiez à la main un adhérent pour qu'il devienne administrateur.

login	mdp	nomAdherent	prenomAdherent	email	dateAdhesion	numCategorie	isAdmin
jcvd	karaté	VAN DAMME	Jean-Claude	jean-claude.van-damme@yopmail.com	1960-10-18	2	0
jupiter	brigitte	MACRON	Emmanuel	emmanuel.macron@yopmail.com	1977-12-21	1	0
lafleche	12345	SELLA	Philippe	philippe.sella@yopmail.com	2022-10-05	3	0
musclor	grrr	CHABAL	Sébastien	sebastien.chabal@yopmail.com	2022-10-05	2	0
rico	delprado	DI MECO	Eric	eric.di-meco@yopmail.com	2022-10-09	3	0
speedy	azerty	DUPONT	Antoine	antoine.dupont@yopmail.com	2022-10-03	1	0
theboss	jojo	MOSCATO	Vincent	vincent.moscato@yopmail.com	2022-10-03	1	1

Remarque : pour le moment, nous sauvegardons les mots de passe « en clair ». Ce n'est pas du tout une bonne pratique. Nous verrons plus tard comment corriger cela.

3. Modifiez la classe `Adherent` pour qu'elle soit adaptée à ce nouvel attribut. En particulier, vous ajouterez cette fonction spécifique :

```
public function isAdmin() {return $this->isAdmin == 1;}
```

4. Modifiez la méthode `addAdherent` pour que par défaut, l'ajout d'un nouvel adhérent affecte 0 au champ `isAdmin` de la table de données.
5. On souhaiterait que seuls les adhérents non administrateurs soient affichés dans la liste des adhérents, car ce sont de fait les seuls qui seront amenés à réserver des livres. Les administrateurs ne sont là que pour gérer les tâches d'administrateur. Pour créer cette différenciation d'affichage de façon générique, voici une possibilité :

- a. Dans la classe `Objet`, créez une méthode d'instance

```
public function affichable() {return true;}
```

qui retourne `true` par défaut. Ainsi, par défaut, tous les objets seront « affichables ».

- b. Dans la classe `Adherent`, redéfinissez cette méthode de façon que seuls les non administrateurs soient affichables :

```
public function affichable() {return !$this->isAdmin;}
```

- c. Dans la méthode `lireObjets` de la classe `ControleurObjet`, conditionnez l'affichage de l'objet courant au fait qu'il soit affichable.
 - d. Pour vérifier que votre code fonctionne, changez à la main dans `phpMyadmin` plusieurs administrateurs en les passant `admin`, et vérifiez que le site ne les affiche plus. Puis revenez à la normale (1 administrateur).

Exercice 2 – formulaire de connexion au site

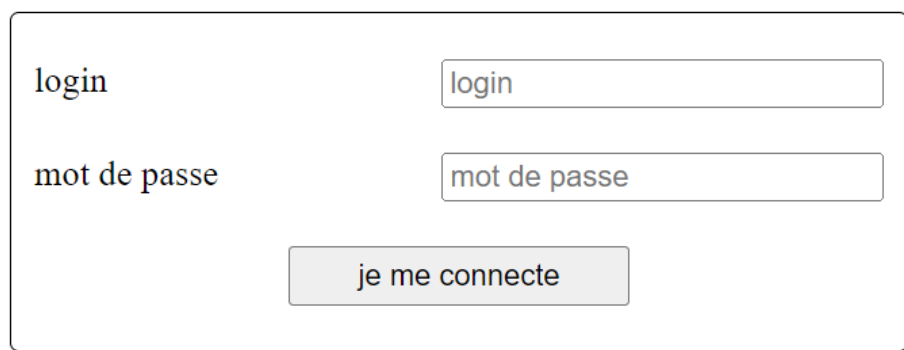
Le problème est toujours le même : pour le moment, tous les adhérents disposent de super-privilèges normalement réservés aux administrateurs.

Nous allons y remédier en mettant en place des **sessions** :

- un internaute ne pourra entrer sur le site qu'en se connectant via un formulaire de connexion (ou bien, s'il n'a pas de compte, il devra en créer un pour ensuite se connecter) ;
- une fois connecté, il est reconnu par son login, et repéré comme étant ou non administrateur ;
- il faudra alors différencier les vues (un admin verra plus de choses qu'un adhérent lambda) ;
- enfin, il faudra penser à la fonctionnalité de déconnexion.

Pour le moment on ne va pas ajouter la fonctionnalité de création de compte. Nous nous contentons de la connexion pour un utilisateur existant.

1. Créez, dans le répertoire `vue`, une vue `formulaireConnexion.html` qui produira l'affichage suivant (ne cherchez pas forcément à le styliser) :



The image shows a login form with a light gray border and a subtle drop shadow. It contains two text input fields. The first field is labeled 'login' and the second is labeled 'mot de passe'. Below these fields is a single button labeled 'je me connecte'.

Le formulaire sera associé à une action « `connecterAdherent` » du contrôleur.

```
<form action="routeur.php" method="get">
  <input type="hidden" name="contrôleur" value="contrôleurAdherent">
  <input type="hidden" name="action" value="connecterAdherent">
```

2. Créez, dans `ContrôleurAdherent`, l'action `afficherFormulaireConnexion` qui appellera ce formulaire de connexion.
ATTENTION : cette action ne doit pas incorporer le menu !
3. Transférez, et testez l'appel de votre formulaire par l'url :
`routeur.php?contrôleur=contrôleurAdherent&action=afficherFormulaireConnexion`

login

login

mot de passe

mot de passe

je me connecte

Bibliothèque 2022

Si votre routeur est bien conçu, le remplissage et l'envoi du formulaire n'aboutiront qu'à la liste des adhérents puisque la méthode `connecterAdherent` du `controleurAdherent` n'est pas encore codée. Ce sera l'objet de l'exercice 5.

Exercice 3 – Mise en place des sessions

Les sessions sont un mécanisme qui permet de personnaliser la connexion au site. Pour mettre en place une session, nous allons ajouter une couche supplémentaire au routeur : la véritable porte d'entrée du site sera un fichier `index.php`, qui aura pour rôle d'ouvrir une session (ou bien de maintenir la session existante) et de passer le relais au `routeur.php`.

1. Déplacez le fichier `routeur.php` (qui est en fait un contrôleur général) dans le répertoire `controleur`. Créez à la racine le fichier `index.php` codé ainsi :

```
1 <?php
2 session_start();
3 // affichage de contrôle du tableau $_SESSION
4 // echo "<pre>session courante : <br>"; print_r($_SESSION); echo "</pre>";
5 require_once("controleur/routeur.php");
6 ?>
7
```

TP12\ex3\index.php 4:6 CRLF UTF-8 PHP GitHub Git (0)

La commande `session_start()` réactive la session existante, ou bien en ouvre une autre. Elle permet d'accéder au tableau `$_SESSION`, qui va contenir les renseignements intéressants sur la session en cours (c'est à nous de le remplir).

Comme c'est le fichier `index.php` qui gère la session, **tout doit maintenant passer par lui** (alors qu'avant la porte d'entrée était `routeur.php`). **DONC :**

2. Partout où on appelait `routeur.php`, remplacez maintenant par `index.php` :
 - dans les liens des vues d'affichage (utilisateur et voiture),
 - dans le `ControleurObjet`, méthode `lireObjets`,
 - dans certains contrôleurs (`Livre` et `Auteur`)
 - dans le menu,
 - dans les `<form action="routeur.php" method="get">` des formulaires.

Vérifiez que tout fonctionne.

Remarque : le fichier `index.php` se lance automatiquement, donc l'url TP9/ex3 suffit.

3. A des fins de contrôle, et provisoirement, nous allons demander à `index.php`, juste après l'instruction `session_start()`, d'afficher l'état actuel du tableau `$_SESSION` par un affichage classique de tableau. Décommentez l'affichage en question dans `index.php`. Testez.

session courante :
Array
(
)

auteur	adhérent	livre	nationalité	genre	catégorie
Auteur 1	définir les nationalités	supprimer	modifier	détails	
Auteur 2	définir les nationalités	supprimer	modifier	détails	
Auteur 3	définir les nationalités	supprimer	modifier	détails	
Auteur 4	définir les nationalités	supprimer	modifier	détails	
Auteur 5	définir les nationalités	supprimer	modifier	détails	
Auteur 6	définir les nationalités	supprimer	modifier	détails	
Auteur 7	définir les nationalités	supprimer	modifier	détails	
Auteur 8	définir les nationalités	supprimer	modifier	détails	
Auteur 9	définir les nationalités	supprimer	modifier	détails	
Auteur 10	définir les nationalités	supprimer	modifier	détails	
Auteur 11	définir les nationalités	supprimer	modifier	détails	
Auteur 12	définir les nationalités	supprimer	modifier	détails	
Auteur 13	définir les nationalités	supprimer	modifier	détails	
Auteur 14	définir les nationalités	supprimer	modifier	détails	
Auteur 15	définir les nationalités	supprimer	modifier	détails	
Auteur 26	définir les nationalités	supprimer	modifier	détails	

Bibliothèque 2022

Exercice 4 – Méthodes pratiques pour les sessions, et modification du routeur

Nous allons créer un fichier de fonctionnalités pratiques en rapport avec les sessions et avec le rôle de la personne qui se connecte au site.

Ce fichier sera un fichier de type Modèle, appelé `session.php`. Créez donc un fichier `modele/session.php`. Ce fichier définira une classe `Session` qui contiendra plusieurs méthodes statiques, à coder dans les questions suivantes.

Remarque : c'est un fichier modèle à part des autres : il n'héritera pas de la classe `Modele`.

```
<?php
class Session {

    public static function userConnected() {

    }

    public static function adminConnected() {

    }

    public static function userConnecting() {

    }

}
```

1. Codez la méthode

```
public static function userConnected()
```

qui dit si oui ou non un adhérent est connecté. Elle retourne donc un booléen qui dit si la variable `$_SESSION["login"]` existe.

2. Codez la méthode

```
public static function adminConnected()
```

qui dit si oui ou non l'administrateur est connecté. Elle retourne donc un booléen qui dit si les variables `$_SESSION["login"]` et `$_SESSION["isAdmin"]` existent, et si `$_SESSION["isAdmin"]` soit égale à 1.

3. Codez la méthode

```
public static function userConnecting()
```

qui dit si oui ou non un adhérent est en train de se connecter. Elle retourne donc un booléen qui dit si le formulaire de connexion a été rempli et renvoyé, en d'autres termes si l'action passée dans l'url existe, et si cette action est `connecterAdherent`.

4. Insérez le fichier `session.php` au niveau du routeur.

5. Un nouvel algorithme pour le routeur : on va maintenant repenser le routeur selon l'algorithme suivant

```
// si aucun utilisateur n'est connecté,  
// et si aucun utilisateur n'est en train de se connecter,  
// alors l'action est "afficherFormulaireConnexion"  
// et le contrôleur est "controleurUtilisateur"  
// sinon  
// - si un contrôleur correct est passé dans l'url, alors on l'utilise.  
// - si une action correcte est passée dans l'url, alors on l'utilise  
// enfin on lance l'action du contrôleur
```

Modifiez le code du routeur pour qu'il agisse selon cet algorithme.

Si en lançant l'url suivante

```
https://...TP9/ex4/index.php
```

vous n'arrivez pas au formulaire de connexion au site, alors vous avez encore du travail 😊

Par contre, si vous arrivez au formulaire de connexion, et si vous le remplissez avec un login et un mot de passe existant, et si votre routeur réagit bien, vous devez obtenir la liste des adhérents...

En effet la méthode `controleurAdherent::connecterAdherent` est appelée par le formulaire mais n'a pas encore été codée. C'est l'un des objectifs de l'exercice 5.

Donc on continue...

Exercice 5 – La méthode `connecterAdherent` du `ControleurAdherent`

1. Une méthode pratique du modèle

Codez, dans le fichier `modele/adherent.php`, une méthode

```
public static function checkMDP($l, $m)
```

qui retourne `true` si un adhérent de login `$l` et de mot de passe `$m` existe dans la base de données, et `false` sinon. Pour cela :

- on récupère le tableau des adhérents dont le login est `$l` et le mot de passe est `$m`.
- si le tableau résultat est de taille 1, alors on renvoie `true`, sinon on renvoie `false`.

2. Codez la fonction `connecterAdherent` du `ControleurAdherent`. Cette fonction va permettre de :

- a. récupérer le login et le mot de passe en provenance du formulaire de connexion,
- b. vérifier s'il existe dans la base un adhérent correspondant à ce login, et si son mot de passe est le même que celui du formulaire (on utilisera `checkMDP`).
- c. si ce n'est pas le cas, on lance la fonction `afficherFormulaireConnexion`
- d. si c'est le cas :
 - affecter à `$_SESSION["login"]` la valeur du login transmis,
 - récupérer l'adhérent dont le login est le login transmis,
 - affecter à `$_SESSION["isAdmin"]` la valeur `isAdmin()` de cet adhérent,
 - insérer les vues habituelles et la vue `lireObjet.php`.

Testez la connexion avec un adhérent. Si vous voulez en tester un autre, il va falloir à la main supprimer les cookies, notamment le cookie `PHPSESSID` qui permet de « suivre » l'adhérent connecté.

Pour cela, il suffit de nettoyer les cookies de l'historique de navigation et de relancer la requête `TP9/ex5/index.php`

Faites-le et reconnectez-vous avec un autre adhérent.

Exercice 6 – La méthode `deconnecterAdherent` du `controleurAdherent`

Il est bien évident qu'un adhérent de base n'a pas à supprimer lui-même les cookies pour se connecter. On va donc mettre en place une fonctionnalité de déconnexion, et un lien qui permet de lancer cette déconnexion.

1. Codez maintenant, dans le `controleurAdherent`, la méthode suivante, qui permet de détruire la session, et de redemander le formulaire de connexion :

```
public static function deconnecterAdherent() {  
    session_unset();  
    session_destroy();  
    setcookie(session_name(), '', time()-1);  
    self::afficherFormulaireConnexion();  
}
```

- `session_unset()` vide le tableau `$_SESSION`
- `session_destroy()` supprime, SUR LE SERVEUR, le fichier de données associé à cette session (car chez le client, la seule donnée stockée est le cookie d'identifiant de session)
- `setcookie(session_name(), '', time()-1)` demande au client de supprimer le cookie de session (sans garantie, car le serveur ne contrôle pas le comportement du client en matière de cookies)

ATTENTION, ces 3 actions sont nécessaires : `session_destroy()` ne suffit pas.

2. On va maintenant créer un affichage personnalisé : si un adhérent est connecté, alors on lui affiche un message de bienvenue, ainsi qu'un lien de déconnexion.

On modifiera légèrement le fichier `menu.html`. Il va maintenant contenir une instruction PHP. Donc il va falloir systématiquement renommer `menu.php` partout où on avait écrit `menu.html`...

L'instruction PHP en question : afficher un message de bienvenue qui incorporera la valeur de `$_SESSION["login"]` et un lien vers l'action `deconnecterAdherent`.

3. Une fois que vous aurez fait quelques essais, vous retirerez aussi l'affichage systématique du tableau `$_SESSION` au niveau du fichier `index.php`. Testez.

Si vous en êtes là, vous avez déjà bien avancé !

Maintenant que l'on peut se connecter et se déconnecter, il faut SECURISER le site. Cela peut prendre deux formes, dont une seule est vraiment efficace :

- Proposer des vues adaptées : un menu administrateur, et un menu adhérent basique, bien différents, avec des liens proposés qui dépendent du rôle de la personne.

Mais attention, les liens ne sont pas les seules portes d'entrées vers les fonctionnalités du contrôleur : on peut directement agir via l'url en court-circuitant les liens du menu. Il est donc essentiel de :

- Sécuriser les actions du contrôleur, qui ne seront déclenchées que si la personne connectée a les autorisations correspondantes.

Exercice 7 – Différenciation des menus

1. Les adhérents non administrateurs auront un menu adapté, l'administrateur aura le menu actuel. Renommez `menu.php` en `menuAdmin.php`, et créez `menuAdherent.php` avec 3 liens :
 - la liste des livres,
 - la liste des auteurs,
 - le profil personnel de l'utilisateur connecté.
2. Dans chacune des méthodes des deux contrôleurs, il y a une insertion du menu. Maintenant, il va falloir insérer soit le `menuUser`, soit le `menuAdmin`.

- a. Dans la classe `Session`, écrivez une méthode

```
public static function urlMenu() {  
  
}
```

qui retourne l'url du menu, à savoir :

- `vue/menuAdmin.php` si l'adhérent connecté est administrateur
 - `vue/menuUser.php` si l'adhérent connecté n'est pas administrateur
- b. Partout où le menu était auparavant naïvement inséré, changer l'instruction pour que la bonne url du menu soit maintenant utilisée, selon le statut de l'adhérent connecté.
 - c. Vérifiez que tout fonctionne, en vous connectant soit comme un administrateur, soit comme un adhérent non administrateur.



Exemple de connexion non administrateur



Exemple de connexion administrateur

ATTENTION ! Le menu « non administrateur » est bien adapté, mais personnaliser les menus n'est qu'une action cosmétique et superficielle. Même si ce n'est pas directement accessible par son menu, un non administrateur peut très bien par l'url accéder aux fonctionnalités de création, ce qui est anormal. Pour s'en convaincre, il suffit de changer à la main dans l'url l'action du `controleurAdherent` en `afficherFormulaireCreationObjet`.

Ceci prouve que les contrôles doivent absolument être effectués au niveau des actions du contrôleur. Certaines actions seront réservées à l'administrateur. Pour sécuriser au mieux les actions, il convient de les factoriser au maximum (pour centraliser la sécurisation).

- Le thème du TP10 : GENERISATION SUITE ET FIN
- Le thème du TP 11 : SECURISER LE SITE