Authentification

Licence Informatique 3ème année

Alexandre Niveau — Jean-Marc Lecarpentier

Authentification

Notes de cours

- **Authentification**
  - Stockage des mots de passe
  - Connexion et déconnexion
  - Restriction des accès

Travail personnel

Objectifs

Appliquer les mécanismes du stockage de mot de passe, de l'authentification, et de la restriction d'accès.

Exercice 1 — Bases de l'authentification

Cette archive contient un fichier `comptes.php`, qui déclare simplement une liste de comptes utilisateur (nom, login, mot de passe, statut) sous forme de tableaux (un compte = un tableau associatif). Les mots de passe ont été hachés avec l'algorithme `bcrypt`, comme montré en cours. La liste contient quatre comptes ; le mot de passe des utilisateurs `toto` et `testeur` est « `azerty` », celui des deux autres est « `1234` ».

Nous allons utiliser ces comptes déjà créés pour faire un mini-site proposant de se connecter/déconnecter, avec en plus une partie réservée aux internautes ayant le statut admin. Le résultat final peut être vu et testé ici : vous pouvez constater que le message de la page d'accueil change si la personne est connectée, et que seul Toto

Dupont a accès à la page d'administration. Il n'y a rien d'intéressant sur aucune des pages — seul le mécanisme d'authentification et de droit d'accès nous intéresse dans cet exercice !

❦ Implémentation directe des fonctionnalités

1. Créer un script `connexion.php` qui inclut `comptes.php`, et qui affiche un formulaire de connexion (login et mot de passe). La soumission du formulaire doit mener à la même page.
2. Le script doit analyser les données reçues : si le login existe dans la base et que le mot de passe correspond, mettre le compte dans une variable de session, par exemple `$_SESSION['user']`. Vous pouvez mettre un feedback comme dans l'exemple si vous voulez, mais ce n'est pas la priorité de l'exercice.
3. À présent, créer un autre script `index.php` pour la page d'accueil. Le contenu de la page doit être personnalisé si la personne est connectée (par ex. « Bienvenue, Jean-Michel ! »).
4. Ajouter aux deux pages un menu de navigation avec un lien vers la page d'accueil et un lien vers la page de connexion/déconnexion.
5. Modifier `connexion.php` pour que, si la personne est connectée, ce ne soit pas un formulaire de connexion mais un bouton de déconnexion qui soit affiché. Le traitement de la déconnexion est simple : la variable de session est supprimée. Vérifier que la connexion/déconnexion fonctionne correctement, et notamment que la page d'accueil s'adapte.
6. Créer un script `admin.php` pour simuler une partie admin : la page doit afficher quelque chose comme « partie admin » si la personne est connectée **et** a le statut d'admin, et doit afficher un message d'erreur comme « accès interdit » sinon. Tester les différents cas.
7. Mettre à jour les menus :
   a. Ajouter un menu à la page d'admin, avec des liens vers les trois pages.
   b. Ajouter un lien vers la page d'admin dans le menu des deux autres pages, mais **attention** : le lien devra apparaître **uniquement** aux admins.
8. Tester que tout fonctionne correctement avec la checklist suivante :
   - si je suis déconnecté·e…
     - ☐ la page d'accueil affiche un message générique et un menu avec deux liens.
     - ☐ la page de connexion affiche un formulaire de connexion et un menu avec deux liens.
     - ☐ la page d'admin affiche un message d'erreur et un menu avec deux liens.
   - si je suis connecté·e en tant que `martine`…
     - ☐ la page d'accueil affiche un message personnalisé et un menu avec deux liens.
     - ☐ la page de connexion affiche un bouton de déconnexion et un menu avec deux liens.

        - ☐ la page d'admin affiche un message d'erreur et un menu avec deux liens.
    - si je suis connecté·e en tant que `toto`…
        - ☐ la page d'accueil affiche un message personnalisé et un menu avec trois liens.
        - ☐ la page de connexion affiche un bouton de déconnexion et un menu avec trois liens.
        - ☐ la page d'admin affiche un message spécifique et un menu avec trois liens.

❦ Vers un code plus propre et maintenable

Gestion centralisée de l'authentification

À présent que le site fonctionne, on va essayer de nettoyer un peu le code, afin qu'il soit plus modulaire.

Tels quels, les différents scripts fonctionnent indépendamment, tout en partageant un certain nombre de « conventions », en l'occurrence le principe de fonctionnement de la connexion, et le nom de la variable de session associée. Ça peut poser problème lorsqu'il faudra faire évoluer le site, car il est facile d'introduire accidentellement des incompatibilités. Pour commencer, nous allons encapsuler les aspects techniques de l'authentification dans une classe, qui offrira à nos scripts une API plus directe.

1. Créer une classe `AuthenticationManager` dans un fichier `AuthenticationManager.php`. Son constructeur devra prendre en paramètre une liste de comptes, et elle devra avoir les méthodes suivantes :
    - `connectUser($login, $password)`, qui gère la connexion (vérifie si le couple login/password est correct et stocke le compte correspondant en session), et renvoie true si la connexion a réussi (et false sinon)
    - `isUserConnected()` qui indique si l'internaute est connecté·e ou non
    - `isAdminConnected()` qui indique si l'internaute est connecté·e avec le statut admin
    - `getUserName()` qui renvoie le nom de l'internaute connecté·e ; si l'internaute n'est pas connecté·e, une exception doit être levée
    - `disconnectUser()` qui déconnecte l'internaute
2. Modifier les trois scripts du site pour qu'ils créent une instance d'`AuthenticationManager`, et l'utilisent autant que nécessaire : il ne doit plus rester trace de `$_SESSION['user']` (ou l'équivalent que vous avez choisi) dans ces scripts — c'est l'`AuthenticationManager` qui « a la main » sur ce point.

Gestion centralisée de l'affichage

Un autre problème avec le site est que — à moins que vous n'ayez bien fait les

choses — la logique (code PHP) et l'affichage (code HTML) ne sont pas bien séparés, et qu'il y a du code HTML qui est répété à plusieurs endroits. Nous allons maintenant améliorer ça, en centralisant la gestion de l'affichage dans une classe dédiée. Après ça, les trois scripts du site ne contiendront plus que de la logique (donc en l'occurrence quasiment rien…)

1. Vérifier que vos pages sont bien valides W3C.
2. Créer une nouvelle classe `Displayer`, qui va se charger de tout ce qui concerne l'affichage du site, de façon centralisée. Le `Displayer` va avoir besoin de faire varier l'affichage en fonction du statut de l'internaute (connecté·e ou non, admin ou non) : pour cela, il utilisera une instance de `AuthenticationManager`, passée en paramètre à son constructeur.

   Le `Displayer` aura en particulier les trois méthodes suivantes :

   - `displayMainPage()`
   - `displayConnectionPage()`
   - `displayAdminPage()`

   Vous pouvez bien sûr ajouter d'autres méthodes si vous le jugez nécessaire — l'idée étant d'éviter autant que possible la répétition de code entre les trois méthodes `displayXYZ`.
3. Modifier les trois scripts du site pour qu'ils utilisent le `Displayer`. Il ne devrait plus rester grand'chose dans les scripts (sauf, notamment, la récupération des données du formulaire de connexion). Vérifiez que les trois pages sont toujours valides.
4. Certaines méthodes du `Displayer` font des choix en fonction du statut de l'internaute (connecté·e ou non). Cela peut vite devenir compliqué de s'y retrouver, et de vérifier que les accès autorisés sont bien conformes à ce que l'on veut. Modifier le Displayer pour en faire une classe abstraite ou une interface, avec trois implémentations, `PublicDisplayer` (pour les internautes non connecté·es), `PrivateDisplayer` (pour les internautes connecté·es) et `AdminDisplayer` (pour les admins). Ces trois classes peuvent s'hériter entre elles si vous pensez que c'est utile.

NB: cet exercice est à visée pédagogique ; les choix architecturaux présentés ne sont pas forcément adaptés à n'importe quel site — l'objectif est de vous faire réfléchir à la modularité, et pratiquer. On ne fera pas toujours les mêmes choix dans la suite du cours