

**Exercice 1 :** reprenons l'exercice 1 de la série précédente

The screenshot shows a web interface for posting a message. It has a title 'Poster un message' and a form with three input fields: 'Login :', 'Mot de passe :', and 'Message :'. Below these is a blue button labeled 'Envoyer votre msg'. Below the form is a section titled 'Les messages déjà postés' containing two entries. The first entry is by 'alain' with the message 'Bien venu dans ce forum'. The second entry is by 'david' with the message 'Bonjour'.

| Poster un message                                |                          |
|--|--------------------------|
| Login :  | <input type="text"/>     |
| Mot de passe :                                   | <input type="password"/> |
| Message :  | <input type="text"/>     |
| <input type="button" value="Envoyer votre msg"/> |                          |

  

| Les messages déjà postés |                         |
|--------------------------|-------------------------|
| alain :                  | Bien venu dans ce forum |
| david :                  | Bonjour                 |

Changez de place avec votre voisin et demandez lui d'effectuer des attaques qui permettent de :

- Ecrire un message sans être inscrit dans la BDD et sans avoir de mot de passe
- Récupérer le noms de la BD
- Récupérer le nom des tables
- Récupérer le nom des colonnes de la table qui contient les mots de passes
- Récupérer le login et mot de passe d'Alain.
- Polluez le forum avec un pop-up JS « bonjour » qui s'affiche à chaque fois en utilisant le login et mdp d'Alain.
- Rendre le forum inaccessible en utilisant une attaque CSS qui rend invisible tout le contenu de la page

Bien entendu, celui qui fait l'attaque n'a pas accès aux fichiers sources de son camarade. Il ne peut utiliser que la page du navigateur pour réaliser ses attaques.

Corrigez maintenant votre code en utilisant des requêtes préparées et en bloquant toute attaque XSS puis passez une dernière fois la main à votre camarade pour qu'il essaye les attaques précédentes.

**Exercice 2 :** Il s'agit de faire évoluer l'exercice 1 avec un système d'authentification par session. L'utilisateur arrive sur cette interface

Se connecter

Login :

Mot de passe :

OK

Si l'identifiant ou mot de passe n'est pas correct on affiche un message d'erreur comme ci-dessous

Se connecter

Login :

Mot de passe :

OK

Erreurs détectées

Login ou mdp non valide

Si l'authentification réussit on affiche l'interface suivante :

Poster un message

Message :

Envoyer votre message

Les messages déjà postés

alain :

Bien Venu [82]

david :

Bonjour [81]

Vous remarquez que l'on insère plus son pseudo si l'on souhaite poster un message étant donné que si on est arrivé à cette page c'est grâce à notre authentification via la page authentication.php. Il suffit donc d'écrire un message et de cliquer sur « envoyer » pour que le message apparaisse dans le forum précédé du nom de l'utilisateur qui l'a envoyé.

Pour quitter le forum il suffit de cliquer sur le bouton « quitter le forum » qui détruit la session.

Notez également que vous si vous ouvrez votre fichier avec deux navigateurs différents (FireFox et Chrome par exemple) alors cela simule deux sessions différentes. Vous pouvez donc bien tester votre site.

Nous allons adapter l'architecture MVC de l'exercice 1 afin qu'elle réponde aux exigences de ce forum. **Attention à ne mettre aucun espace avant tous les <?php de tous vos fichiers autrement vous aurez une erreur liée à l'utilisation de session\_start.**

### Modification de la vue :

Créez dans le répertoire vue le fichier gabaritConnect.php qui contient tout le code Html de l'interface de saisie du login et mot de passe. Ce gabarit contient également comme unique instruction php <?php echo \$contenu; ?> qui servira à remplir le contenu du gabarit d'une manière similaire à ce qui a été fait pour le fichier gabarit.php des deux exercices précédents

Ajoutez dans le fichier vue.php de la vue la fonction afficherConnect() qui affecte une chaîne vide à \$contenu puis inclut le fichier gabaritConnect.php.

Modifiez la fonction afficherErreur pour qu'elle prenne trois paramètres au lieu d'un seul : afficherErreur(\$erreur,\$gabarit,\$retour). Elle affecte à la variable Php \$contenu (qui sert à remplir le gabarit) le code HTML du formulaire qui permet d'afficher le message d'erreur puis teste si \$retour=true en que cas elle affecte en plus à \$contenu le code HTML pour afficher un lien vers la page forum.php. Le client pourra cliquer sur ce lien pour revenir à la page d'accueil du forum et voir les discussions. Cette fonction se termine par require\_once(\$gabarit) qui aura comme conséquence d'inclure le fichier dont le nom est dans la variable \$gabarit.

### Modification du contrôleur

Ajoutez ensuite dans le fichier controleur.php du contrôleur la fonction CtlAcceuilConnect() qui appelle la vue via la fonction afficherConnect(); que vous avez créée à l'étape précédente.

Ajoutez la fonction CtlDeconnect() qui détruit la session puis appelle la vue de connexion afficherConnect()

Ajoutez la fonction CtlConnect(\$login,\$mdp) qui récupère dans la variable \$pseudo le résultat de l'appel de la fonction checkUser(\$login,\$mdp) du modèle.

- Si le résultat égale null elle crée une variable de session \$\_SESSION['nom'] qui contiendra le nom de celui qui s'est connecté (c'est-à-dire \$pseudo[0]→nom) puis appelle CtlAcceuil() du contrôleur qui va lancer une demande d'affichage des discussions.
- Sinon elle génère une exception avec le message : Login ou mdp non valide

Modifiez la fonction CtlAjouterMessage pour qu'elle ne prenne en entrée que deux paramètres : CtlAjouterMessage(\$nom,\$message) et teste uniquement si le message n'est pas empty :

- Si le test est négatif : elle appelle ajouterMessage(\$nom,\$message) du modèle ;
- Sinon génère une exception avec le message : Message vide.

Modifiez la fonction CtlErreur(\$erreur) pour qu'elle teste l'existence de la variable de session \$\_SESSION['nom'] et :

- si le test réussit elle appelle afficherErreur(\$erreur,'gabarit.php',true); de la vue
- sinon elle appelle afficherErreur(\$erreur,'gabaritConnect.php',false); de la vue

### **Modification du contrôleur frontal**

On commence par un session\_start() pour manipuler les variables de session. On met ensuite dans le bloc try un test d'existence de la variable de session \$\_SESSION['nom'] qui contient le nom de l'utilisateur qui est déjà en session :

- si le test réussit cela signifie qu'un utilisateur est déjà en session. On fait alors ces tests :
  - si le bouton envoyer est posté, on charge \$nom avec \$\_SESSION['nom'], et \$msg avec le message posté puis on appelle CtlAjouterMessage(\$nom,\$msg); du modèle suivi de CtlAcceuil(); du contrôleur
  - sinon si le bouton déconnecté est posté on appelle CtlDeconnect() du contrôleur
  - sinon on appelle CtlAcceuil() du contrôleur
- sinon cela signifie qu'il n'y a pas d'utilisateur en session. On fait alors ces tests :
  - si le bouton connexion est posté on charge dans \$user le login posté et dans \$mdp le mot de passe posté puis on appelle CtlConnect(\$user,\$mdp) du contrôleur
  - sinon cela signifie qu'on vient d'arriver sur la page et on appelle ctlAcceuilConnect() du contrôleur ;

Après le bloc try on maintient le même contenu pour le block catch de l'exercice 1.