Université Grenoble Alpes - IUT2 - Département informatique

R3.01 - Programmation WEB - 2024

Objectifs du TP:

- Utiliser la notion de session.
- Revoir l'achitecture MVC (sans modèle dans cet exemple). Manipuler les accesseurs explicites (getter/setter) (facultatif)

Cet exercice vous montre comment utiliser la notion de session pour réaliser une opération de 'login' dans une application. Cette application est composée des vues suivantes :

1. Mise en place d'une session

 login.html: l'utilisateur s'identifie avec son login et son mot de passe. • main.php: la fenêtre de l'application, elle n'a qu'un seul paramètre: \$login le nom de login de l'utilisateur qui s'affiche dans l'entête et dans la page principale.

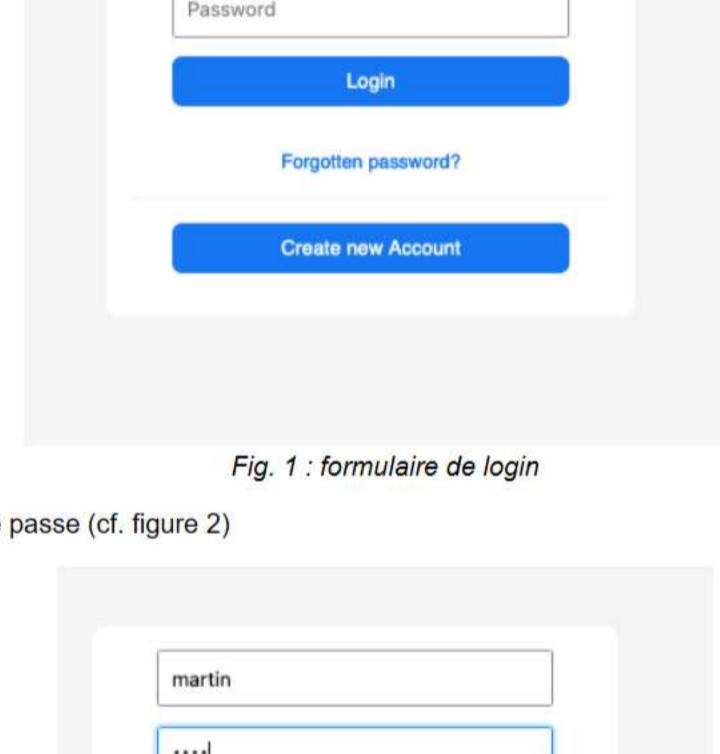
 not_implemented.html: affiche un message qui indique que la fonctionalité n'est pas réalisée. Cette application n'a que deux contrôleurs :

• index.php: c'est le contrôleur principal. Il doit afficher la vue de login si l'on n'est pas connecté, ou bien envoyer sur la page principale de l'application. Ce fichier est également le contrôleur principal et l'application. du formulaire de login. C'est lui qui teste le mot de passe de l'utilisateur. Il renvoit à la page not_implemented.html pour les actions non développées.

• logout.php : détruit la session, permet donc de se déconnecter et de retourner à la vue de login. Le fonctionnement est le suivant : lorsque l'on active la première fois login.php on voit apparaître le formulaire de login (cf. figure 1).

Your login here

TP 10: login avec une session (avec solutions)





▼ Réponse :

fichier par:

include("view/main.php");

//var_dump(\$_SESSION);

// C'est terminé !

exit(0);

switch(\$submit) {

if (isset(\$_SESSION['login'])) {

\$login = \$_SESSION['login'];

include("view/main.php");

// Recupère le login pour la vue

// Charge la page de l'application et termine

// On n'est pas déjà connecté, on examine ce qu'on doit faire

▼ Réponse :

▼ Réponse : Pour ne pas rendre visible à l'utilisateur par l'URL son mot de passe.

Parce que dans un modèle MVC, toute action sur une vue doit toujours passer par un contrôleur. Ici, on a simplifié avec un lien direct sur une autre vue.

 Est-ce que le masquage du mot de passe en entrée (input) avec l'usage de POST est suffisant pour la sécurité de ce mot de passe ? ▼ Réponse :

Complétez le code du contrôleur index.php. Il faut réaliser les opérations suivantes :

• Du coté du PHP (index.php), comment peut-on savoir si l'utilisateur à cliqué sur le bouton "login" ou "new" ?

Il faut regarder la valeur (value) associée au nom (name) "submit" des boutons. Il y a deux valeurs possibles : "login" et "new".

les échanges HTTP et donc utiliser le protocole HTTPS.

isset) si cette entrée existe dans la session (i.e. l'entrée 'login' existe dans '\$_SESSION'), nous permet de savoir si l'utilisateur est connecté ou non. Il est inutile de stocker le mot de passe dans la session. Si l'utilisateur est connecté, alors il faut récupérer son nom de login depuis la session, et choisir la vue principale. Le choix d'une vue se fait simplement en chargant le

vient du formulaire ou si l'on active directement le fichier "index.php". Pour mieux comprendre ce qu'il se passe affichez le contenu des variables \$_POST et \$_SESSION, sachant que ce tableau n'existe qu'après avoir ouvert la session. Losque l'on active directement "index.php", il faut tester si une session est ouverte, et dans ce cas afficher la page principale (cas traité ci-dessus), sinon afficher la page de login.

Si l'utilisateur n'est pas connecté alors on lui présente le formulaire de login.

données. Il faut ensuite lui présenter la page principale.

Pensez à traiter explicitemeent (de préférence par un <u>switch</u>) les 3 cas suivants :

L'utilisateur a cliqué sur "new" : il faut lui afficher la page "not_implemented.html".

// Si la session connait le login alors on s'est déjà connecté, on va sur la page principale

// Ouverture de la session session_start();

L'utilisateur a cliqué sur "login" : il faut vérifier son nom de login et mot de passe. Ces deux valeurs sont placées "en dur" dans le code. Dans la réalité, ils sont dans une base de

L'utilisateur n'a pas utilisé le formulaire : c'est le cas lorsque l'on active directement "index.php". Attention : avec l'utilisation de la méthode "post", on ne voit pas dans l'URL si l'on

case 'login': // Vérification du login et mot de passe if ((\$login -- 'martin') && (\$password -- '1234')) { \$_SESSION['login'] = \$login; // Charge le page de l'application include("view/main.php"); else { // Retourne sur le login include("view/login.html"); case 'new': // Fonctionalité non implementée include('view/not_implemented.html'); default: // Envoit sur le login include("view/login.html"); ?> Complétez le contrôleur logout.php. Ildoit simplement détruire la session par : session_destroy();

protected

function __construct(string \$id, string \$libelle, float \$montant HT) {

// Taux atuel de la TVA mis comme une constante

// Identifiant unique de l'article

pour interdire l'accès aux attributs. Le contrôle d'accès est alors reporté dans les méthodes de type setter et getter. Par exemple, examinez le code suivant :

et

<?php

const TVA = 20.0;

class Article {

private \$id;

// Description

private \$libelle;

// Définition d'un article

// Montant sans la taxe

// Construction d'un article

\$this->libelle = \$libelle;

function getLibelle() : string {

return \$this->libelle;

Il permet de donner accès à trois attributs :

libelle

Montant_HT

<?php

const TVA = 20.0;

class Article {

private \$id;

// Description

private \$libelle;

// Définition d'un article

// Montant sans la taxe

// Construction d'un article

\$this->libelle = \$libelle;

\$this->montant_HT = \$montant_HT;

if (\$property === "montant_TTC") {

public function __set(string \$property, \$value)

function __get(string \$property) {

private \$montant_HT;

\$this->id = \$id;

// Taux atuel de la TVA mis comme une constante

// Identifiant unique de l'article

et

\$this->montant_HT = \$montant_HT;

private \$montant_HT;

\$this->id = \$id;

function getMontant_HT() : float { return \$this->montant_HT; function getMontant_TTC() : float {

```
Montant_TTC
     <?php
    // Test de la classe Article
    // Lancer le test en mode shell par : php Article.test.php
    require_once("Article.class.php");
    // Crée un nouvel article
    $article = new Article("68733560","Coffret de douilles 91 pièces",79.90);
    printf("Article : %s\n",$article->getLibelle());
    printf("Prix TTC : %s\n", $article->getMontant_TTC());
Si le nombre des attributs à contrôler est important, cela augmente en proportion le nombre d'accesseurs (dont le code est trivial). Certains IDE produisent pour vous ces méthodes de manière
automatique. Le langage PHP a une autre solution : implémenter la méthode
  __get
, le getter universel. (nb : une solution equivalente existe aussi en Javascript). Examinez les codes suivant :
```

```
throw new Exception("reading property '$property' is not allowed");
 ?>
<?php
// Test de la classe Article
// Lancer le test en mode shell par : php Article.test.php
require_once("Article.class.php");
// Crée un nouvel article
$article = new Article("68733560","Coffret de douilles 91 pièces",79.90);
// Affichage de cet objet
printf("Article : %s\n",$article->libelle);
printf("Prix TTC : %s\n", $article->montant_TTC);
 ?>
```

```
. Cette méthode permet alors d'interdire d'ajouter des attributs aux objets après leur création par new, comme c'est le cas par défaut de tout objet.
On vous propose maintenant de completer le code de cette classe en ajoutant la notion de quantité de produit.
À faire :
```

• Compléter le code pour permettre de prendre en compte la quantité d'un article. On doit pouvoir changer cette quantité qui par défaut doit être à 1. Il faut aussi contrôler que cette nouvelle quantité soit strictement positive. Il faut finalement revoir le calcul du montant TTC pour tenir compte de cette quantité. Utilisez le setter __set

• Récupérer le code data/accesseur.zip, le comprendre et le tester. En particulier, constater que l'on peut accéder aux attributs 'libelle', 'montant_HT, mais pas 'id'.

pour mettre en place ce contrôle. ▼ Solution : Télécharger le fichier : data/ORM getSolution.zip

Mettre en place un login pour une application Web.

L'utilisateur doit alors entrer son identifiant de login et son mot de passe (cf. figure 2) Login Forgotten password? Create new Account Fig. 2 : entrée du login et mot de passe Si le login et le mot de passe sont corrects alors la page de l'application est affichée avec le nom de login de l'utilisateur (cf. figure 3). This is the main windows of your app martin

du header (cf. figure 3 en haut à droite). L'application revient alors à l'affichage du login. A faire: Téléchargez les codes pour cet exercice depuis le fichier data/login.zip var dump(\$ POST); Examinez le formulaire HTML de la vue login.html. Quel fichier PHP va recevoir et traiter les données de ce formulaire ?

Le nom de login est conservé dans la session, ce qui permet de passer l'étape de login lorsque l'on active à nouveau index.php. Pour fermer la session, l'utilisateur doit activer le bouton "Logout" Attention : cette application respecte les principes MVC, il ne faut donc pas activer directement les fichiers des vues (dans le répertoire view) • Ouvrez le fichier index.php qui sera déclanché par le formulaire. Placez dans ce fichier un simple affichage du contenu de la variable super globale \$_POST avec la commande : ▼ Réponse : Cette information est dans l'attribut action du formulaire. Il s'agit donc du contrôleur index.php. Pourquoi l'URL du lien de "Forgotten password?" contient le répertoire "view" ? ▼ Réponse : Parce que ce qui active (par un include) ces vues sont les contrôleurs qui sont dans le répertoire principal. Toutes les URL relatives, sont relatives à l'URL qui à été activée, c'est à dire l'URL du contrôleur "index.php". Pourquoi le lien de "Forgotten password?" n'est pas une bonne solution ?

 Pourquoi la méthode post est utilisée dans ce formulaire ? Quand l'utilisateur entre son mot de passe, il n'est pas affiché à l'écran. Qu'est-ce qui conditionne ce fonctionnement (dans le HTML et/ou dans le CSS) ? ▼ Réponse : Ce n'est pas dans le CSS, c'est dans le HTML : c'est le type "password" de la balise input.

Non : ce n'est pas suffisant, car avec POST le mot de passe voyage en clair sur le réseau dans le corps du message HTTP. Pour un envoi sécurisé du mot de passe, il faut encoder

 Récupérer les 3 valeurs du formulaire. Démarrer la session par session_start(); o On propose de stocker dans la session la variable \$login. Pour stocker dans la session une variable il suffit de modifier le tableau super global \$_SESSION: \$_SESSION['login'] = \$login; Le fait de savoir (par

▼ Solution : <?php //var_dump(\$_POST); // Récupérations des données du formulaire \$login = \$_POST['login'] ?? ''; \$password = \$_POST['password'] ?? ''; \$submit = \$_POST['submit'] ?? '';

puis afficher le formulaire de login. Attention : il faut tout de même ouvrir la session pour pouvoir la détruire ! ▼ Solution : <?php // Ouverture de la session session_start(); // Fermeture session_destroy(); // Retourne au login include("view/login.html"); ?> 2. Les accesseurs : getters et setters Bonus Le modèle Orienté Objet permet de représenter des entités avec un 'savoir' (les attributs), et un 'savoir faire' (les méthodes). Ce modèle permet aussi de limiter les dépendances entre les autres parties d'un logiciel, en contrôlant l'accès aux attributs et aux méthodes. Par exemple, il est important que les attributs d'un objet soient privés, et que les accès à ces attributs soient contrôlés par des méthodes, qu'on appelle des getters, et setters, ou plus généralement des accesseurs. Le fait d'utiliser des méthodes pour accéder aux attributs permet non seulement d'en contrôler l'accès mais aussi d'ajouter une couche d'abstraction et permettre éventuellement de déclencher du code lors de l'accès à un attribut. Cet aspect est important en PHP, où les programmes doivent s'exécuter dans le temps le plus court, car tant que le programme PHP coté serveur n'est pas terminé, l'utilisateur doit attendre coté client (i.e. le navigateur). Il faut donc s'attacher à limiter les calculs au strict nécessaire lors de la production d'une ressource Web. Dans le langage PHP, il existe, comme en Java ou C++, les qualificatifs private

?>

return \$this->montant_HT * (1+TVA/100);

. Il interdit l'accès à l'attribut id , simplement en n'implémentant pas le getter. Notez également que cette classe donne accès à l'attribut calculé Montant_TTC , qui n'existe pas de manière statique dans l'objet, mais qui est calculé à la demande. L'accès aux attributs de l'extérieur de la classe, doit alors se faire avec les méthodes accesseurs prévues. Par exemple :

return \$this->montant_HT * (1+TVA/100); } else if (isset(\$this->\$property) && (\$property != 'id')) { return \$this->\$property; } else {

function __construct(string \$id,string \$libelle,float \$montant_HT) {

L'accesseur universel __get

s'occupe du contrôle de tous les accès aux attributs en une seule méthode. Dans l'usage de la classe, la syntaxe de l'accès aux attributs est simplement l'accès aux attributs comme s'ils étaient

publics, alors que leur accès déclenche bien le code d'un accesseur comme dans la version précédente.

De manière symétrique, il existe en PHP le moyen d'avoir un contrôle d'accès en écriture sur les attributs. Il s'agit de la méthode

Sujet de TP Programmation WEB (PHP).