**Une image contenant texte, clipart

Description générée automatiquementConcepteur Développeur d’Applications**

**Evaluation Rattrapage**

**PHP/HTML/CSS/JS : Application Web**

**Durée : 2 h**

**Documents autorisés : Oui**

**Internet autorisé : Oui**

**Consignes :**

Ce document est à compléter et à mettre à la racine de votre espace réseau dans le dossier **EVAL\_PHP\_Rattr\_[Prénom]**

L’évaluation porte sur **vos connaissances en matière de développement web**.

* Les réponses doivent être précises. Mettre des extraits courts de code (uniquement ceux qui répondent à la question), préciser le fichier et la méthode s’il y a lieu.

**Sujet :**

# Quels sont les points techniques qui permettent de passer de :

## La page connexion au catalogue

Lors de la validation du formulaire (FormConnexion.php), les infos des inputs sont envoyées à l’actionConnextion par la méthode POST

  <form action="index.php?page=ActionConnexion" method="POST">

Les points techniques se situent dans actionConnexion.php et sont :

* La vérification de l’existence du compte utilisateur passé par l’intermédiaire de la variable $\_POST

$uti =  UtilisateursManager::getList(null, ['adresseMail' => $\_POST['adresseMail']]);

if ($uti != null) {

* La vérification du mot de passe

if ($uti[0]->getMotDePasse() == crypte($\_POST['motDePasse'])) {

* La redirection vers la page catalogue

 header("location:index.php?page=Catalogue");

* De plus on initialisera la variable $\_SESSION afin de garder les infos de l’utilisateur connecté

 $\_SESSION['utilisateur'] = $uti[0];

## La liste des articles au formulaire de modification

Dans le fichier ListeArticle.php, lors du clique sur le bouton Modifier, on appelle le fichier FormArticle.php grace à sa route à laquelle on précise l’action à effectuer et l’id de l’article que l’on souhaite modifier.

echo '<div class="caseListe"> <a href="index.php?page=FormArticles&mode=Modifier&id='.$unObjet->getIdArticle().'"><i class="fas fa-pen"></i></a></div>';

Dans le fichier FormArticle, on récupère l’action à effectuer dans la variable $\_GET

$mode = $\_GET['mode'];

ainsi que l’article dont l’id est passé par l’intermédiaire de la variable $\_GET

if (isset($\_GET['id'])) {

    $elm = ArticlesManager::findById($\_GET['id']);

Et l’on affiche le formulaire en renseignant les inputs grâce aux infos récupérées dans $elm. Exemple pour le libelle de l’article :

echo '<div class="caseForm donneeForm"><input type="text" '.$disabled .'value="'.$elm->getLibelleArticle().'" name=LibelleArticle pattern="'.$regex["\*"].'"></div>';

## Comment s’effectue le retour

Le retour vers la liste des articles se fait par le bouton retour qui appelle la route du fichier ListeArticles.php grace à cette ligne :

<div><a href="index.php?page=ListeArticles"><button type="button"><i class="fas fa-sign-out-alt fa-rotate-180"></i></button></a></div>

La route appelée par cette commande est la suivante :

"ListeArticles" => ["PHP/VIEW/LISTE/", "ListeArticles", "Liste Articles", 2, false],

# La liste des articles me fournit un IdTypeArticle, comment récupérer un texte ?

Pour récupérer le texte correspondant à l’idTypeArticle reçu, il faut appeler la fonction findById de TypesArticleManager à laquelle on donne l’id du type d’article en paramètre. Cette fonction va nous retourner un objet de type TypesArticle auquel on appliquera la fonction ToString de la classe TypesArticles afin de ne récupérer que le Libellé du type de l’article.

public function toString()

    {

        return $this->getLibelleTypeArticle();

    }

Cette action est réalisée grâce à la ligne suivante :

echo '<div class="caseListe left color5  gridCase">'.TypesArticlesManager::findById($unObjet->getIdTypeArticle())->ToString().'</div>';

elle se situe dans le fichier ListeArticles.php.

# Dans le formulaire article

## Comment faire apparaitre l’image ?

Dans le formulaire article (FormArticle.php), la ligne qui nous permet de faire apparaitre l’images est la suivante :

echo '<div class="caseForm donneeForm"><img src="./IMG/'.$elm->getPhotos().'" /></div>';

Elle fait appel à la balise HTML img qui permet d’afficher une image, dans la source on définit le chemin du dossier contenant les fichiers image qui est «./IMG/ »auquel on concatène le nom du fichier contenu en base de donnée et que l’on retrouve dans l’objet Article stocké dans la variable $elm. Pour accéder à l’attribut photo de cet objet, on utilise le getter getPhoto() ;

## Comment transformer l’input TypeArticle pour avoir une liste déroulante ?

La transformation de l’input TypeArticle en liste déroulante, se fait grace à cette ligne :

echo '<div class="caseForm donneeForm">'.creerSelect($elm->getIdTypeArticle(),"TypesArticles",["LibelleTypeArticle"],$disabled).'</div>';

## Quels fichiers, quelles méthodes entre en jeu ?

Pour créer cette liste déroulante, plusieurs méthodes et fichiers entrent en jeu.

Tout d’abord le fichier Outils.php. Dans ce fichier, nous allons utiliser la méthode « creerSelect» qui nous permet de créer la liste déroulante proprement dite. Nous allons lui passer plusieurs paramètres :

* L’idTypeArticle à selectionner dans le cas ou on est en mode « Editer, Modifier, Supprimer »
* La Table contenant les informations de la liste déroulante
* Et le nom de la Colonne à afficher dans la liste, ici le libelle
* La variable $disabled initialisée plus haut permet de rendre la liste non modifiable dans les modes « Editer et Supprimer »

Dans cette commande on utilise également la méthode getIdTypeArticle qui se trouve dans le fichier TypesArticles.Class.php qui nous permet de récupérer l’idTypeArticle de l’objet Article contenu dans la variable $elm.

## Quels sont les modifications à apporter pour permettre le changement d’image ?

Pour pouvoir changer l’image plusieurs modifications sont nécessaires. Tout d’abord au niveau du formulaire :

On ajoute l’emplacement pour afficher la photo :

echo '<div class="caseForm donneeForm"><img src="./IMG/'.$elm->getPhotos().'" /></div>';

Ainsi qu’un input de type file qui nous permet de sélectionner un fichier :

<input type="file" name="Photos" '.$disabled.'/></div>';

Afin que les données de cet input soient correctement envoyées par le formulaire, il faut ajouter l’encodage à la balise <form> du formulaire. Cet encodage se réalise grace à cette instruction :

enctype="multipart/form-data"/

Ensuite, nous devons ajouter un fichier Javascript nommé ici FormArticles.js

Dans ce fichier, nous commençons par récupérer l’input de type File et la balise image

inputFile = document.querySelector("input[type=file]");

img = inputFile.parentNode.parentNode.querySelector("img")

On applique un event Listener sur l’input File afin de lancer la fonction choixImage lors du changement de valeur de cet input.

 inputFile.addEventListener("change", choixImage);

On crée un objet File reader qui permet de créer une copie de l’image en mémoire

reader = new FileReader();

            reader.addEventListener("load", chargeImage);

            reader.readAsDataURL(img.file);

La fonction chargeImage permet de charger cette image dans l’input.

function chargeImage(e) {

    img.src = e.target.result;

}

On ajoute un input caché à la DOM du formArticle signalent que l’on a changé la photo :

 modif = document.createElement("input");

            modif.name = "modifImage";

            modif.type = "hidden";

            inputFile.parentNode.appendChild(modif);

# Dans les classes, au niveau des setters, que veut dire ?int ?

Indique que la valeur qui suit peut-être de type int ou NULL.

# Traduction : quels sont les points techniques qui entrent en jeu ? (A tous les niveaux)

Les points techniques pour la traduction sont :

* La table texte de la base de données, contenant les différentes traduction anglais -> francais référencées par leur code texte.
* Au niveau des différents fichiers, pour mettre en œuvre la traduction il faut faire un appel a la fonction texte suivit du code du texte à traduire comme l’exemple suivant :

texte("Deconnexion")

Cette fonction texte se situe dans le fichier outils.php elle recupère la colonne de la table Texte correspondant au codeTexte passé en paramètre selon la variable lang contenu dans la session ($\_Session).

function texte($codeTexte)

{

    $retour=TextesManager::findByCodes($\_SESSION['lang'], $codeTexte);;

    if ($retour==false) return $codeTexte;

    return $retour;

}

La fonction findByCode doit être créée dans le fichier TexteManager.Class.php afin de récupérer les informations en base de données.

* Il faut donc initialiser la variable lang dans la session lors de la connexion d’un utilisateur et la faire varier de ‘fr’ à ‘en’ afin de passer en traduction anglaise.

# Dans quel(s) contexte(s) sert le fichier DAO.Class.php

DAO -> Data Access Object

Ce fichier est appelé à chaque fois qu’un manager veux accéder à la base de donnée pour envoyer ou récupérer des données. Les méthodes qu’il contient sont les seules pouvant accéder à la base de donnée.

# Utilisateurs : A quel endroit intervenir pour forcer le rôle et le mot de passe ?

Pour forcer le rôle et le mot de passe d’un utilisateur, il faut intervenir dans le fichier ActionUtilisateurs.php.

Le rôle est forcé par cette ligne :

$elm->setRole(1);

Dans ce cas il est forcé à 1 donc utilisateur.

Le mot de passe est forcé par cette ligne :

$elm->setMotDePasse(crypte($elm->getNom().$elm->getPrenom()));

La méthode crypte se trouvant dans le fier Outils.php permet de crypter le mot de passe avant de l’envoyer en base de données.

# Panier Utilisateur : Quelles sont les actions à mener pour récupérer les lignes du panier du client ?

Dans le fichier ListeLignesPaniers.php on commence d’abord par récupérer le panier du client connecte grâce à cette instruction :

$cmd = PaniersManager::getList(null, ["idClient" => $\_SESSION['utilisateur']->getIdUtilisateur()]);

Ensuite on récupère les lignes correspondantes à ce panier grâce à l’instruction suivante :

    $objets = LignesPaniersManager::getList(null, ["idPanier" => $cmd[0]->getIdPanier()]);

La variable $objets contiendra donc les lignes du panier du client connecte.

# Comment gérer les listes vides ? A quel(s) niveau(x) ?

On gère les listes vides dans :

* Listepanier.php on récupère la liste des paniers et si elle est vide on affiche le message suivant « Pas de paniers actuellement disponibles »

$objets = PaniersManager::getList();

if (!$objets) echo '<div class="caseListe titreListe center">Pas de paniers actuellement disponibles </div>';

si non on affiche les paniers des clients

* ListeLignesPaniers.php si le panier du clientn’a pas de ligne, on affiche le message « Pas de panier en cours »

$cmd = PaniersManager::getList(null, ["idClient" => $\_SESSION['utilisateur']->getIdUtilisateur()]);

/\*  On recupere les lignes de la Panier \*/

if (!$cmd)

    echo '<div class="gridListe center">Pas de panier en cours</div>';

else {

si non on affiche les lignes du panier

# Comment fixer le footer en bas de la page et permettre de voir toute la page ?

Pour positionner le footer en bas de la page, il faut intervenir au niveau du fichier style.css et definir pour le footer la position fixed et lui indiquer qu’il doit etre fixé en bas grace pour ce faire on ajoutera les lignes suivantes :

position: fixed;

    bottom:0;

on ajoutera une marge en bas de la page afin que le contenu ne soit pas caché par le footer. Dans le fichier style.css on ajoutera l’instruction suivante a la balise mail :

    padding-bottom: 20vh;

# Présentation sous forme de Grid :

## Où intervenir, comment le paramétrer ?

Pour la présentation de base sous forme de grid, les déclarations se situent dans le fiechier grid.css on peut modifier les couleurs de bordure etc ou créer de nouvelles grids

## Comment obtenir une case qui fusionne 2 cases adjacentes ?

Pour obtenir une case qui fusionne 2 cases adjacentes, il faut appliquer la classe css « gridcolumns-span-2 » qui se trouve dans le fichier grid.css a la div html

# Gestion des erreurs : que faut il faire pour mettre en place une gestion d’erreur ? A quel endroit ?

Il faut créer la route correspondante au code d’erreur dans le fichier index.php et appelant le fichier Erreur.php

 "404" => ["PHP/VIEW/GENERAL/", "Erreur", "404", 0, false],

Dans cette application pour la gestion de l’erreur 404, toujours dans le fichier index.php on fait appel à cette route si la page recu en paramètre dans la variable $\_GET n’existe pas :

f (isset($\_GET["page"])) {

    $page = $\_GET["page"];

    if (isset($routes[$page])) {

        AfficherPage($routes[$page]);

    } else {

        AfficherPage($routes["404"]);

    }

Il faut également ajouter les textes de l’erreur à la table texte si l’on souhaite que la traduction soit prise en charge.