**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

* ActionConnexion.php dans PHP/CONTROLLER/ACTION traite les informations de connections, contrôle l’email et le mot de passe
* $uti =  UtilisateursManager::getList(null, ['adresseMail' => $\_POST['adresseMail']]);
* if ($uti != null) {
* if ($uti[0]->getMotDePasse() == crypte($\_POST['motDePasse']))
* La redirection avec le header
* header("location:index.php?page=Catalogue");
* La route dans Index.php charge la page catalogue avec
  + La présence de la route
* "Catalogue" => ["PHP/VIEW/LISTE/", "ListeCatalogue", "Catalogue", 1, false]
  + La méthode AfficherPage du fichier Outils.php dans PHP/Controller
* AfficherPage($routes[$page]);
  + Le fichier ListeCatalogue.php dans PHP/VIEW/LISTE

## La liste des articles au formulaire de modification

* Dans le fichier ListeArticles.php dans PHP/VIEW/Liste contient des redirections en fonctions des actions.

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

La redirection concerne la modification, la route à établir est FormArticles, les paramètres mode et id permettent de préciser à FormArticles.php dans PHP/VIEW/Form les variations à mettre en place en cas de modification

* La route dans Index.php charge la page le formulaire article avec
  + La présence de la route
* "FormArticles" => ["PHP/VIEW/FORM/", "FormArticles", "Formulaire Articles", 2, false]
  + La méthode AfficherPage du fichier Outils.php dans PHP/Controller
* AfficherPage($routes[$page]);
  + Le fichier FormArticles.php dans PHP/VIEW/Form

## Comment s’effectue le retour

* Le bouton qui permet le retour contient un lien vers la page ListeArticles
* <a href="index.php?page=ListeArticles">
* La route dans Index.php charge la page la liste des articles avec
  + La présence de la route
* "ListeArticles" => ["PHP/VIEW/LISTE/", "ListeArticles", "Liste Articles", 2, false]
  + La méthode AfficherPage du fichier Outils.php dans PHP/Controller
* AfficherPage($routes[$page]);
  + Le fichier ListeArticles.php dans PHP/VIEW/Liste

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

Dans ListeArticles.php dans PHP/VIEW/LISTE

TypesArticlesManager::findById($unObjet->getIdTypeArticle())->ToString()

On recherche l’objet TypeArticle à partir de son id

On appelle la methode toString de typeArticle dans TypeArticle.Class.php dans PHP/Controller/Classe qui renvoi un texte.

return $this->getLibelleTypeArticle();

# Dans le formulaire article

## Comment faire apparaitre l’image ?

<img src="./IMG/'.$elm->getPhotos().'" />

$elm est un objet de type Article, on appelle la méthode getPhotos de Articles.Class.php dans PHP/CONTROLLER/CLASSE pour récupérer le nom de la photo.

On définit la source de la balise image

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

creerSelect($elm->getIdTypeArticle(),"TypesArticles",["LibelleTypeArticle"],$disabled)

La méthode créerSelect permet de créer une liste déroulante à partir des paramètres entrés

## Quels fichiers, quelles méthodes entre en jeu ?

* méthode CreerSelect dans Outils.php dans PHP/Controller, qui crée un select à partir des paramètres
* méthode getAttributes de la classe TypeArticles dans PHP/Controller/Classe, qui ramène les différentes colonnes de la table
* méthode getList de la classe TypeArticlesManager.class.php dans PHP/Model/Manager, qui ramène les types d’articles contenu dans la table
* méthode AppelGet dans Outils.php dans PHP/Controller, qui déclenche des méthodes tel que getLibelleTypeArticle pour récupérer les informations à afficher

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

* Ajouter un input de type File pour permettre la sélection d’un nouveau fichier
* <input type="file" name="Photos" '.$disabled.'/>
* Ajouter un input caché pour garder le nom de l’ancienne image
* <input type="hidden" name="AnciennePhotos" value="'.$elm->getPhotos().'">
* Mettre en place un javascript qui permet de repérer la sélection du fichier
* Dans JS/ FormArticles.js
* inputFile.addEventListener("change", choixImage);
* Vérifier le type de fichier
* if (!imageType.test(fichier.type))
* Charger l’image en mémoire
* reader = new FileReader();
* reader.addEventListener("load", chargeImage);
* reader.readAsDataURL(img.file);
* Modifier la source de la balise img
* img.src = e.target.result;
* Changer l’encryptage du form dans FormArticles.php dans PHP/VIEW/FORM
* <form class="GridForm" action="index.php?page=ActionArticles&mode='.$\_GET['mode'].'" method="post" enctype="multipart/form-data"/>
* Récupérer le fichier dans ActionArticles.php dans PHP/Controller/Action
* move\_uploaded\_file($\_FILES['Photos']['tmp\_name'], 'IMG/' . $leNom)
* Mettre le nouveau nom de fichier dans la base
* $elm->setPhotos(chargerImage());

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

Que le paramètre d’entrée doit être un entier ou null

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

* Dans la base de données, mise en place et alimentation de la table texte
* Création d’un manager dédié à l’exploitation de cette table, TexteManager.Class.php dans PHP/Model/Manager
* Création d’une méthode d’appel courte à la traduction dans Outils.php dans PHP/Controller
* function texte($codeTexte)
* Appel à cette méthode lorsque l’on veut traduire un texte dans les tous les fichiers qui permettent de l’affichage, par exemple
* texte(« DescriptionArticle »)

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

* Le fichier DAO sert chaque fois qu’il est nécessaire d’accéder à la base de données
* En écriture pour tous les add, update, delete de tous les managers PHP/Models/Manager
* En lecture pour tous les getList, findById de tous les managers PHP/Models/Manager
* Ainsi que pour la connexion à la base de données, méthode Init de DBConnect.Class.php dans PHP/Models/Manager

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

Dans ActionsUtilisateur.php dans PHP/Controller/Action

$elm->setRole(1);

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

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

Dans ListeLignesPaniers.php dans PHP/VIEW/LISTE

* On récupère le panier de l’utilisateur
* $cmd = PaniersManager::getList(null, ["idClient" => $\_SESSION['utilisateur']->getIdUtilisateur()]);
* On récupère les lignes du panier
* $objets = *LignesPaniersManager*::getList(null, ["idPanier" => $cmd[0]->getIdPanier()]);

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

* Dans DAO.Class.php on teste si la requête à ramener des lignes
* if (!$q) return false;
* Si ce n’est pas le cas, on renvoi False
* Dans le manager correspondant (exemple LignesPaniersManager dans PHP/MODEL/MANAGER) dans le getListe, on renvoi l’information reçue du DAO (on propage de False en cas de liste vide)
* return DAO::select(
* Dans le fichier de gestion de la liste (exemple ListeLignesPaniers.php dans PHP/VIEW/LISTE) on ajoute un test au retour de la fonction getList
* $cmd = PaniersManager::getList(null, ["idClient" => $\_SESSION['utilisateur']->getIdUtilisateur()]);
* if (!$cmd)

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

Dans Style.css dans CSS, dans la partie footer{}

 position: fixed;

    bottom:0;

dans la partie main{}

    padding-bottom: 20vh;

# Présentation sous forme de Grid :

## Où intervenir, comment le paramétrer ?

* Dans le fichier CSS correspondant ici grid.css
* On crée une classe (exemple grid-col-7) que l’on appliquera au html
* display : grid;
* permet de gérer l’élément sous forme de grid
* grid-template-columns : 2fr 2fr 2fr 2fr 0.5fr 0.5fr 0.5fr;
* définit le nombre de colonne et la proportion des colonnes les unes par rapport aux autres
* grid-gap: 1em;
* donne l’écart entre 2 lignes de la grid

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

* Dans le css, dans la partie qui définit l’élément (exemple : grid-columns-span-2)
* grid-column:1/3;

ou

* grid-column : span 2;
* donne la proportion de la case par rapport au nombre de cases

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

* Créer un fichier Erreur.php dans PHP/VIEW/GENERAL
* Créer une route pour l’atteindre
* "Erreur" => ["PHP/VIEW/GENERAL/", "Erreur", "titreErreur", 0, false]
* Faire appel à la route en cas d’erreur (exemple ActionConnexion dans PHP/Controller/Action)
* header("location:index.php?page=Erreur&cible=Connexion&codeErreur=MdpIncorrect");
* Mettre à jour la table texte pour avoir le libellé de l’erreur (par exemple)
* INSERT INTO `textes`(`codeTexte`, `FR`, `EN`) VALUES
* ('erreurAjouter','L\'ajout a échoué', 'Add failed'),
* ('erreurModifier', 'La modification a échoué','Update failed'),
* ('erreurSupprimer', 'La suppression a échoué','Delete failed') ;
* Le paramètre cible permet de connaitre la route après affichage de l'erreur