**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.
* La réponse 1.1 contient la réponse attendue pour exemple.

**Sujet :**

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

## La page connexion au catalogue (Exemple)

* L’actionConnexion 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 : on indique dans la redirection que l’on doit rester dans l’index et on passe en paramètre dans « page » le fichier qui doit être appelé, dans le « mode » si c’est un ajout, une modif ou une suppression et l’id de l’objet concerné.

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 index.php dans $routes:

"FormPaniers" => ["PHP/VIEW/FORM/", "FormPaniers", "Formulaire Paniers", 2, false],

L’appel de la fonction pour afficher la page dans l’index.php :

if (isset($routes[$page])) {

        AfficherPage($routes[$page]);

## Comment s’effectue le retour

* Si l’utilisateur change d’avis et ne veut plus modifier il peut être redirigé vers une autre page grâce à un bouton « retour »

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

* Si l’utilisateur a modifié un article le retour se fait grâce à un header qui se situe dans l’ActionArticles :
* header("location:index.php?page=ListeArticles");

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

On récupère le texte en faisant un findById en donnant l’IdTypeArticle en paramètre, ce qui nous retourne toutes les données liées à cet id.

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

# Dans le formulaire article

## Comment faire apparaitre l’image ?

Dans le formulaire article, on fait apparaitre l’image en appelant le getter photos à la suite du chemin dans le src de la balise img.

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

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

Pour avoir une liste déroulante pour le TypeArticle, on appelle la fonction « creerSelect », qui se situe dans le fichier Outils.php et on spécifie que dans l’input grâce au paramètre $nomColonnes, on ne doit voir que le libelle du typeArticle.

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

## Quels fichiers, quelles méthodes entre en jeu ?

Les fichiers et méthodes qui entrent en jeu pour le formulaire article sont :

FormArticles.php pour l’affichage du formulaire,

DAO.Class.php qui contient toutes les fonctions qui impactent la base de données

Fonction add :

public static function add($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $requ = "INSERT INTO " . $class . "(";

Fonction update :

public static function update($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $requ = "UPDATE " . $class . " SET ";

Fonction delete :

public static function delete($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $methode = "get" . ucfirst($colonnes[0]);

        return $db->query("DELETE FROM " . $class . " WHERE " . $colonnes[0] . " = " . $obj->**$methode**());

    }

Fonction select :

public static function select(array $nomColonnes, string $table, array $conditions = null, string $orderBy = null, string $limit = null, bool $api = false, bool $debug = false)

    {

        $db = DbConnect::getDb();

        $string = json\_encode($nomColonnes) . $table . json\_encode($conditions) . $orderBy . $limit . $api . $debug;

        if (strpos($string, ";")) {

            return false;

        } else if (!empty($nomColonnes)  && ($table != null && $table != "")) {

            $cols = self::elementSelect($nomColonnes);

            $t = " FROM " . $table;

ArticlesManager.Class.php qui contient des fonctions qui appellent les fonctions du DAO,

public static function add(Articles $obj)

    {

        return DAO::add($obj);

    }

Articles.Class.php qui est la classe POCO qui contient les attributs, getters, setters et constructeur,

class Articles

{

    private $\_IdArticle;

    private $\_LibelleArticle;

    private $\_DescriptionArticle;

    private $\_PrixArticle;

    private $\_Photos;

    private $\_IdTypeArticle;

    private static $\_attributes=["IdArticle","LibelleArticle","DescriptionArticle","PrixArticle","Photos","IdTypeArticle"];

ActionArticles.php qui appelle différentes fonctions du Manager selon le mode,

switch ($\_GET['mode']) {

    case "Ajouter": {

        $elm->setPhotos(chargerImage());

        $elm = ArticlesManager::add($elm);

        break;

    }

Outils.php qui sert pour le créerSelect ainsi que pour l’autoload des classes,

function ChargerClasse($classe)

{

    if (file\_exists("PHP/CONTROLLER/CLASSE/" . $classe . ".Class.php"))

    {

        require "PHP/CONTROLLER/CLASSE/" . $classe . ".Class.php";

    }

    if (file\_exists("PHP/MODEL/MANAGER/" . $classe . ".Class.php"))

    {

        require "PHP/MODEL/MANAGER/" . $classe . ".Class.php";

    }

}

spl\_autoload\_register("ChargerClasse");

L’affichage des pages

function afficherPage($page)

Et les regex pour les inputs.

$regex = [

    "alpha" => "[A-Za-z]{2,}-?[A-Za-z]{2,}",

    "alphaNum" => "[A-Za-z0-9]\*",

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

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

Dans les classes au le ?int au niveau des setters veut dire que la variable peut être *null*:

public function setIdArticle(?int $IdArticle)

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

Pour la traduction on a besoin de connaitre la langue de l’utilisateur, on fait cela grâce à l’index.php :

if (isset($\_GET['lang']) && TextesManager::checkIfLangExist($\_GET['lang'])) {

*// tester si la langue est gérée*

    $\_SESSION['lang'] = $\_GET['lang'];

} else if (isset($\_COOKIE['lang'])) {

    $\_SESSION['lang'] = $\_COOKIE['lang'];

} else {

    $\_SESSION['lang'] = 'FR';

}

On peut également stocker la langue dans un cookie :

setcookie("lang", $\_SESSION['lang'], time() + 36000, '/');

Grâce au TextesManager.Class.php, on peut chercher une langue par le code fourni :

public static function findByCodes($codeLangue,$codeTexte)

    {

        $retour = DAO::select([$codeLangue],"textes",["codeTexte" => $codeTexte],null,null,true);

        if ($retour==false) return false;

         return $retour[0][$codeLangue];

    }

Et aussi voir si la langue existe dans la base de données :

public static function checkIfLangExist($codeLangue)

    {

        $db=DbConnect::getDb();

        $q=$db->prepare("SHOW COLUMNS FROM textes LIKE :codeLangue");

        $q->bindValue(":codeLangue", $codeLangue, PDO::PARAM\_STR);

        $q->execute();

        $results = $q->fetch(PDO::FETCH\_ASSOC);

        return ($results != false); *// renvoi vrai si la requete retourne des données, false sinon*

    }

Dans Outils.php on a une fonction qui permet de retourner le texte dans la langue de l’utilisateur si celle-ci existe bien dans la base de données :

function texte($codeTexte)

{

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

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

    return $retour;

}

Ensuite, on peut appeler cette fonction texte au différents endroits où il y aurait besoin d’avoir une traduction :

echo '<label for=LibelleArticle class="caseForm labelForm">'.texte("LibelleArticle").'</label>';

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

Le fichier DAO.Class.php sert à ajouter, modifier, supprimer ou sélectionner des valeurs en base de donnée.

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

Pour forcer le rôle d’un utilisateur celui-ci est automatiquement alloué et le mot de passe automatiquement crée et sécurisé dans le fichier ActionUtilisateurs.php quand un compte utilisateur est créé ou modifié :

$elm = new Utilisateurs($\_POST);

$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 ?

Pour récupérer les lignes du panier du client, on récupère tout d’abord l’id du client pour récupérer son ou ses panier(s) uniquement et pas ceux des autres clients

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

Si le client ne possède pas de panier, on revoie un message

if (!$cmd)

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

else {

Sinon on récupère la liste des paniers

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

Puis chaque article qui se situe dans le(s) panier(s) de l’utilisateur

foreach ($objets as $unObjet) {

        echo '<div class="caseListe donneeListe">' . ArticlesManager::findById($unObjet->getIdArticle())->toString() . '</div>';

        echo '<div class="caseListe donneeListe">' . $unObjet->getQuantite() . '</div>';

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

        echo '<div class="caseListe"> <a href="index.php?page=FormLignesPaniers&mode=Supprimer&id=' . $unObjet->getIdLignePanier() . '"><i class="fas fa-trash-alt"></i></a></div>';

    }

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

Pour gérer le listes vides on appelle dans par exemple ListePaniers.php la fonction getList du Panier Manager que l’on stocke dans une variable :

$objets = PaniersManager::getList();

Et si cette variable est vide alors on affiche un message indicatif sinon on retourne la liste des paniers :

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

else {

*//Création du template de la grid*

    echo '<div class="grid-col-2 gridListe">';

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

Pour fixer le footer en bas de la page pour permettre de voir toute la page, on utilise la propriété fixed sur celui-ci :

footer{

    background-color: var(--color5);

    color: var(--color1);

    font-size: 0.8em;

    position: fixed;

    bottom:0;

    width: 100%;

    padding: 2em;

}

# Présentation sous forme de Grid :

## Où intervenir, comment le paramétrer ?

Pour une présentation sous forme de grid, il faut le paramétrer dans un fichier CSS puis appeler les propriétés crées dans les différentes balises qui en ont besoins dans « class ».

echo '<div class="grid-col-8 gridListe">';

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

Pour obtenir une case qui fusionne 2 cases adjacentes, il faut créer et utiliser une propriété grid-span-2.

echo '<div class="caseListe titreListe grid-columns-span-2">Information sur les Paniers </div>';

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

Pour la gestion des erreurs, il faut créer un ErreurManager ainsi que ses fonctions notamment la fonction « findByCodes » qui récupèrera les codes erreurs fournis et qui affichera ce code si il est répertorié

<?php

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

*// echo ErreurManager::findByCodes($\_GET['codeErreur']);*

            echo texte($\_GET['codeErreur']);

        }else{

*// echo ErreurManager::findByCodes("inconnue");*

            echo texte("inconnue");

        }

?>

Et qui redirigera l’utilisateur soit sur une page cible, soit sur l’accueil selon le cas

<?php

            if (isset($\_GET['cible'])) {*//si on a une page où retourner, on y va*

                echo '<a href="index.php?page='.$\_GET["cible"].'"><i class="fas fa-sign-out-alt fa-rotate-180"></i></a>';

            }else{*//sinon, go acceuil*

                echo '<a href="index.php?page=accueil"><i class="fas fa-sign-out-alt fa-rotate-180"></i></a>';

            }

?>

# API : (Les extraits de code ne seront pas issus du projet, mais des projets réalisés en cours)

## Quels sont les différentes parties d’une API (nommer les parties et les langages dans lesquels ces parties peuvent être écrites ?

L’API peut être divisée en 2 parties l’une écrite en JavaScript qui fera l’appel AJAX et la seconde qui pourrait être écrite en PHP qui appellera une fonction du manager par exemple :

* Partie JS 1

var getlocation = document.**location**.**href**;

var url = new URL**(**getlocation**)**;

var page = url.**searchParams**.get**(**'page'**)**;

var mode = url.**searchParams**.get**(**'mode'**)**;

* Partie AJAX

if **(**page == 'formVille' && mode == 'voir'**)** {

    var request = new XMLHttpRequest**()**;

*//on met à 'true' pour l'asynchrone*

    request.open**(**'POST', 'index.php?page=ListePersonneApi', true**)**;

    request.setRequestHeader**(**'Content-Type', 'application/x-www-form-urlencoded'**)**;

    idVille = document.getElementsByName**(**'idVille'**)[**0**]**.**value**

    args = "idVille=" + idVille;

    request.send**(**args**)**

* Partie JS 2

request.onreadystatechange = function (event) {

        if **(**this.**readyState** === XMLHttpRequest.**DONE)** {

            if **(**this.**status** === 200**)** {

                reponse = JSON.parse**(**this.**responseText)**;

                reponse.forEach**(**element => {

                    template = document.getElementById**(**"perso"**)**;

                    cible = document.getElementById**(**"personne"**)**;

                    contenu = template.content.cloneNode**(**true**)**;

                    cible.appendChild**(**contenu**)**;

                    pers = document.querySelector**(**".pers"**)**

                    pers.**innerHTML** = element.**nom** + " " + element.**prenom**

                    pers.classList.remove**(**"pers"**)**

                }**)**;

            }

        }

    }

}

* Partie PHP :

<?php

$idVille = $\_POST['idVille'];

$listePersonne = PersonnesManager::getList(null, ['idVille'=>$idVille], null, null, true, false);

echo json\_encode($listePersonne);

## Où mettre l’adresse du serveur d’API ?

On met l’adresse du serveur de l’API en second paramètre dans la fonction open de la requête.

request.open**(**'POST', 'index.php?page=ListeVillesApi', true**)**;

## Comment passer des paramètres à l’API ?

Pour passer des paramètres à l’API, on crée un variable argument avec les paramètres puis on passe cette variable à la requête :

args = "idVille=" + idVille;

request.send**(**args**)**

## Comment passer d’une API asynchrone à une API synchrone ?

Pour passer d’une API asynchrone à une API synchrone, on spécifie cela après l’url de l’API dans la fonction open de la requête (true pour asynchrone, false pour synchrone)

request.open**(**'POST', 'index.php?page=ListeVillesApi', true**)**;