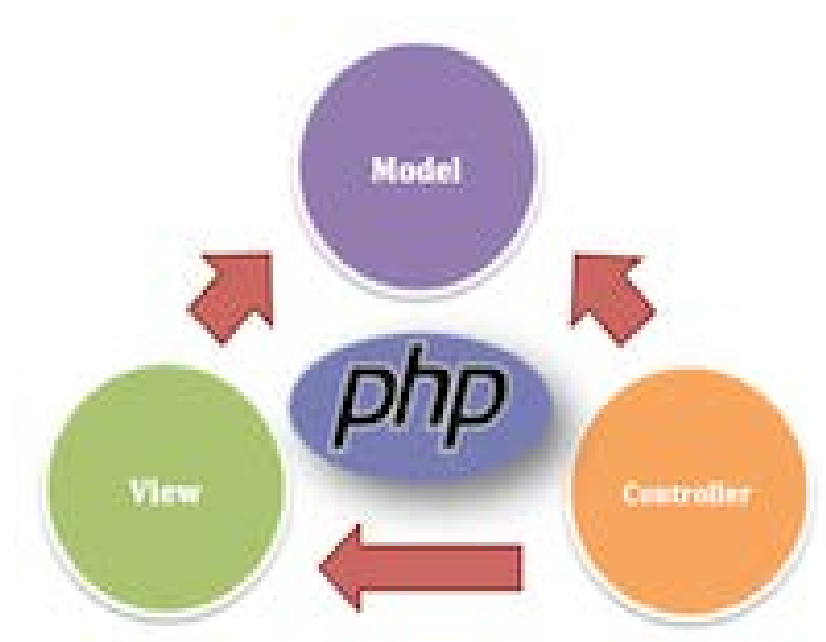


Rapport du projet Mon Blog

Réalisation d'un blog en PHP orienté Objet en utilisant le design pattern MVC



Réalise par : Najoua Bentaher

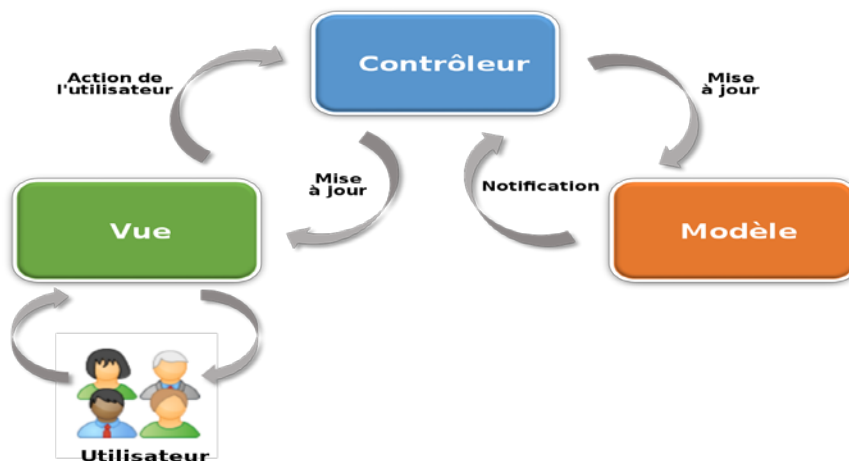
Module : Programmation Oriente
objet en PHP

DCESS : MSIIDE 2023-2025

Enseignant : Khalid
AMECHNOUE

Introduction

Ce rapport documente la création d'un blog en PHP orienté Objet en suivant le modèle de conception MVC (Modèle-Vue-Contrôleur). Le modèle MVC est une approche architecturale qui sépare les différentes responsabilités de développement logiciel, en organisant le code en trois couches distinctes : le Modèle pour la gestion des données et de la logique métier, la Vue pour l'interface utilisateur, et le Contrôleur pour la gestion des interactions entre les deux premières couches.



Les besoins fonctionnels de ce projet sont les suivants :

- Utilisation d'une base de données nommée **monblog** constituée de trois tables : **t_billet**, **t_commentaire** et **t_auteur**.
- Implémentation d'une page d'authentification permettant l'accès au blog.
- Création d'une page **VueAccueil** pour afficher la liste des billets.
- Ajout de la fonctionnalité de création et de publication de nouveaux billets.
- Développement d'une page **VueBillet** pour visualiser un billet spécifique ainsi que les commentaires associés.
- Intégration d'une page **VueErreur** destinée à signaler et à gérer les diverses erreurs survenues lors de l'utilisation du blog.

Base de données

La base de données utilisée est relativement simple, composée de trois tables distinctes. La première table stocke les billets du blog, la deuxième enregistre les commentaires associés à ces billets, et enfin, la troisième table gère l'authentification des auteurs.

La table *t_auteur* :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 aut_id	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/>	2 username	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	3 password	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	4 aut_nom	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)			Modifier Supprimer Plus

☐ Tout cocher Avec la sélection : Parcourir Modifier Supprimer Primaire Unique Index Spatial

La table *t_billet* :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 BIL_ID	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/>	2 BIL_DATE	datetime			Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	3 BIL_TITRE	varchar(100)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	4 BIL_CONTENU	varchar(400)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus

☐ Tout cocher Avec la sélection : Parcourir Modifier Supprimer Primaire Unique Index Spatial T

La table *t_commentaire* :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 COM_ID	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/>	2 COM_DATE	datetime			Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	3 COM_AUTEUR	varchar(100)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	4 COM_CONTENU	varchar(200)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	5 BIL_ID	int(11)			Non	Aucun(e)			Modifier Supprimer Plus

☐ Tout cocher Avec la sélection : Parcourir Modifier Supprimer Primaire Unique Index Spatial T

Cette base de données contient quelques données de test, insérées par le script SQL ci-dessous :

```
insert into T_BILLET(BIL_DATE, BIL_TITRE, BIL_CONTENU) values
(NOW(), 'Premier billet', 'Bonjour monde ! Ceci est le premier billet sur
mon blog.');
```

```
insert into T_BILLET(BIL_DATE, BIL_TITRE, BIL_CONTENU) values
(NOW(), 'Au travail', 'Il faut enrichir ce blog dès maintenant.');
```

```

insert into T_COMMENTAIRE(COM_DATE, COM_AUTEUR, COM_CONTENU, BIL_ID)
values(NOW(), 'A. Nonyme', 'Bravo pour ce début', 1);
insert into T_COMMENTAIRE(COM_DATE, COM_AUTEUR, COM_CONTENU, BIL_ID)
values(NOW(), 'Moi', 'Merci ! Je vais continuer sur ma lancée', 1);

insert into T_AUTEUR(USERNAME, PASSWORD, AUT_NOM) values
('nabentaher46@gmail.com', 'e66448ca10', 'najoua');

```

Page d'authentification

L'accès à la base de données dans la classe abstraite Modele

 *Modele.php*

```

<?php
abstract class Modele {
    private $bdd;
    // Exécute une requête SQL éventuellement paramétrée
    protected function executerRequete($sql, $params = null) {
        if ($params == null) {
            $resultat = $this->getBdd()->query($sql);
        } else {
            $resultat = $this->getBdd()->prepare($sql);
            $resultat->execute($params);
        }
        return $resultat;
    }
    // Renvoie un objet de connexion à la BD en initialisant la
    // connexion au besoin
    private function getBdd() {
        if ($this->bdd == null) {
            // Création de la connexion
            $this->bdd = new
            PDO('mysql:host=localhost;dbname=monblog;charset=utf8',
            'root', '', array(PDO::ATTR_ERRMODE =>
            PDO::ERRMODE_EXCEPTION));
        }
        return $this->bdd;
    }
}
?>

```

Auteur.php

```
<?php
    require_once 'Modele.php';
    class Auteur extends Modele {
        // Renvoie les informations sur un billet
        public function getAuteur($username,$password) {
            $sql = 'select aut_id as id, username, password, aut_nom as nom'
                . ' from t_auteur where username=? and password=?';

            $auteur = $this->executerRequete($sql, array($username,$password));
            if ($auteur->rowCount() == 1)
                return $auteur->fetch(); //Accès à la première ligne de résultat
            else
                return false;
        }
    }
?>
```

Le contrôleur est constitué d'une série d'actions rédigées sous forme de fonctions visant à vérifier l'authentification des auteurs et effectuer d'autres tâches connexes.

ControleurAuteur.php

```
<?php
    require_once '../Modele/Auteur.php';
    session_start();

    class ControleurAuteur {
        private $auteur;

        public function __construct() {
            $this->auteur = new Auteur();
        }
        ##### Méthode pour gérer le processus de connexion
        public function login($username, $password) {
            if(empty($username) || empty($password)) {
                $this->redirectWithError("Vérifiez que vous avez saisi vos données.");
            }

            $username = $this->validate($username);
```

```

        $password = $this->validate($password);
        $auteur = $this->auteur->getAuteur($username, $password);

        if ($auteur) {
            $this->setUserSession($auteur);
            $this->redirectToHome("Logged in successfully!");
        } else {
            $this->redirectWithError("Les données que vous avez saisi
            sont incorrect.");
        }
    }

    ### Méthode privée pour configurer la session de l'utilisateur
    private function setUserSession($auteur) {
        $_SESSION['username'] = $auteur['username'];
        $_SESSION['password'] = $auteur['password'];
        $_SESSION['nom'] = $auteur['nom'];
        $_SESSION['id'] = $auteur['id'];
    }

    private function validate($data) {
        return htmlspecialchars(trim($data));
    }

    ### Méthode privée pour rediriger vers la page de connexion avec
    un message d'erreur
    private function redirectWithError($error) {
        header("Location: ../login.php?error=$error");
        exit();
    }

    ### Méthode privée pour rediriger vers la page d'accueil avec un
    message de succès
    private function redirectToHome($message) {
        header("Location: ../index.php?message=$message");
        exit();
    }
}

##### Vérifie si la méthode HTTP utilisée est POST
$controleur = new ControleurAuteur();
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['uname'] ?? "";
    $password = $_POST['password'] ?? "";

    $controleur->login($username, $password);
}

?>

```

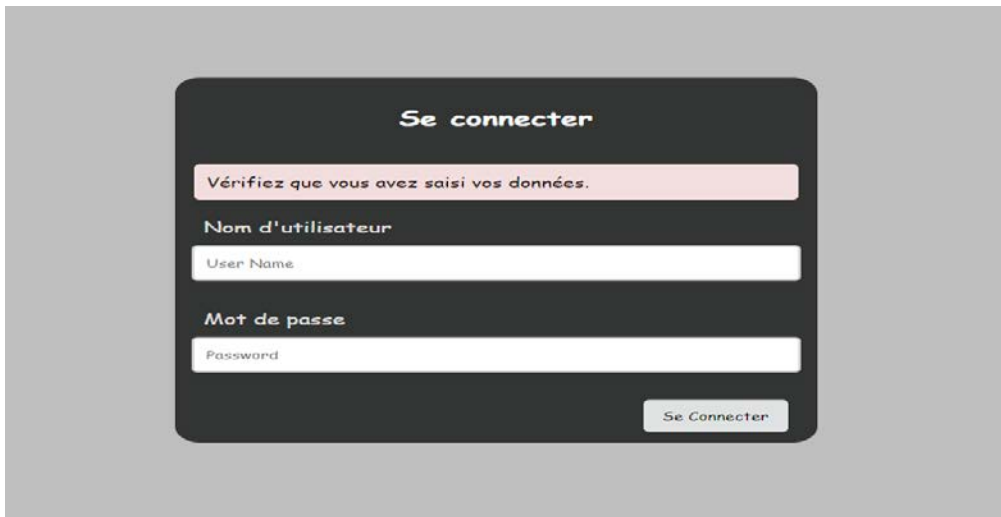
Ensuite, nous avons une nouvelle page appelée **login.php**, conçue pour afficher un formulaire permettant de saisir les informations de connexion et d'accéder à la page d'index du blog.

 *login.php*



The image shows a login form titled "Se connecter". It contains two input fields: "Nom d'utilisateur" (User Name) and "Mot de passe" (Password). Below the fields is a button labeled "Se Connecter".

Dans ce cas, il existe une méthode dans le contrôleur qui vérifie que les champs de saisie ne sont pas vides, et qui retourne un message d'erreur sous forme d'alerte en cas de champ vide.



The image shows the same login form as before, but with an error message displayed at the top: "Vérifiez que vous avez saisi vos données." (Check that you have entered your data). The form fields and the "Se Connecter" button are still present.

Après la saisie des données, une action **login** de connexion est envoyée au contrôleur pour vérifier si les informations saisies existent dans la table **t_auteur**. Si le nom d'utilisateur ou le mot de passe ne sont

pas corrects, un message d'erreur est affiché, sinon il redit vers la page **index** de blog.



Page d'accueil

Une fois que l'auteur est authentifié, la page principale du blog affiche les articles ajoutés à ce blog.

Il y a une classe nommée Billet est responsable de l'accès aux données des articles de blog.

 *Billet.php*

```
<?php
require_once 'Modele.php';
class Billet extends Modele {
    // Renvoie la liste des billets du blog
    public function getBillets() {
        $sql = 'select BIL_ID as id, BIL_DATE as date,'
        . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
        . ' order by BIL_ID desc';
        $billets = $this->executerRequete($sql);
        return $billets;
    }
    // Renvoie les informations sur un billet
    public function getBillet($idBillet) {
```



```

        $sql = 'select BIL_ID as id, BIL_DATE as date,'
        . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
        . ' where BIL_ID=?';

        $billet = $this->executerRequete($sql, array($idBillet));
        if ($billet->rowCount() == 1)
            return $billet->fetch(); // Accès à la première ligne de
                                    // résultat
        else
            throw new Exception("Aucun billet ne correspond à
            l'identifiant '$idBillet'");
    }
}
?>

```

Le contrôleur **Billet** est composé d'une série de méthodes permettant de récupérer les données via la base de données et de les afficher sur la page principale, en se basant sur l'identifiant de chaque billet.

ControleurBillet.php

```

<?php
    require 'Modele/Billet.php';

    // Affiche la liste de tous les billets du blog
    function accueil() {
        $billets = getBillets();
        require 'Vue/vueAccueil.php';
    }

    // Affiche les détails sur un billet
    function billet($idBillet) {
        $billet = getBillet($idBillet);
        $commentaires = getCommentaires($idBillet);

        require 'Vue/vueBillet.php';
    }
}
?>

```

Voici la vue de la page principale, il existe un bouton pour déconnexion, redirige vers la page de l'authentification, ainsi que la liste des billets qui sont stockés dans la table **billet**.

 *VueAccueil.php*



Lorsqu'un billet est cliqué, une page s'affiche, présentant en détail les informations spécifiques à ce billet.

 *VueBillet.php*



L'ajout des commentaires

On souhaite maintenant que l'affichage des détails sur un billet permette d'ajouter un nouveau commentaire. Le remplissage des champs Auteur et Commentaire est obligatoire. Le clic sur le bouton Commenter déclenche l'insertion du commentaire dans la base de données et la réactualisation de la page Web.

Commencer par ajouter à la classe Commentaire une méthode permettant d'insérer un nouveau commentaire dans la BD.

Commentaire.php

```
public function ajouterCommentaire($auteur, $contenu, $idBillet) {
    $sql = 'insert into T_COMMENTAIRE(COM_DATE, COM_AUTEUR,
    COM_CONTENU, BIL_ID)'
    . ' values(?, ?, ?, ?)';
    $date = date('DATE_W3C'); // Récupère la date courante
    $this->executerRequete($sql, array($date, $auteur, $contenu,
    $idBillet));
}
```

Il faut également ajouter au contrôleur une méthode associée à cette action.

Controleur.php

```
// Ajoute un commentaire à un billet
function commenter($auteur, $contenu, $idBillet) {
    // Sauvegarde du commentaire
    $commentaire = ajouterCommentaire($auteur, $contenu, $idBillet);
    // Actualisation de l'affichage du billet
    require 'Vue/vueBillet.php';
}
```

Index.php

```
}else if ($_GET['action'] == 'commenter'){
```

```

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

            $idBillet = $_POST['id'];
            $auteur    = $_POST['auteur'];
            $contenu    = $_POST['contenu'];

            if ($idBillet != 0) {
                commenter($auteur,$contenu,$idBillet);
            }
            else
                throw new Exception("Identifiant de billet non valide");
        }
        else
            throw new Exception("Identifiant de billet non défini");
    }
}

```

On ajoute ensuite à la vue d'un billet le formulaire HTML nécessaire pour saisir un commentaire.

 *VueBillet.php*

```

<form method="post" action="<?=" index.php?action=commenter&id=" .
    $billet['id'] ?>">
    <input id="auteur" name="auteur" type="text" placeholder="Votre pseudo"
    required /><br />
    <textarea id="txtCommentaire" name="contenu" rows="4"
    placeholder="Votre commentaire" required></textarea><br />
    <input type="hidden" name="id" value="<?=" $billet['id'] ?>" />
    <input type="submit" value="Commenter" />
</form>

```

Lorsque vous ajoutez le commentaire, il apparaît de cette manière.



L'affichage des erreurs

Si une erreur survient, il est possible de maintenir la structure de la vue en définissant une vue dédiée nommée **vueErreur.php**. Ensuite, on ajuste le contrôleur pour rendre cette vue en cas d'erreur.

 *Contrôleur.php*

```
<?php

require 'Modele/Modele.php';

// Affiche une erreur
function erreur($msgErreur) {
    require 'Vue/vueErreur.php';
}

?>
```



GitHub

https://github.com/NajouaBentaher/examen_php.git