

ENCADRÉE PAR:
AFEF SLAMA

GÉNÉRATEUR DE CV

cyrine ben chabene
wissal frihi



SOMMAIRE

- 1 introduction
- 2 **CREATION DE COMPTE**
- 3 **LOGIN**
- 4 **CLASSE USER**
- 5 **CLASSE PERSONNE**
- 6 **BASE DE DONNEES**

GÉNÉRATEUR DE CV

L'objectif est de décomposer un modèle de CV HTML et PHP pour une utilisation en ligne. La structure du CV comprend les sections habituelles : information personnelle, compétences, expérience professionnelle et éducation.



CREATION DE COMPTE

- Ce formulaire permet à l'utilisateur de créer un compte en fournissant son nom, son prénom, son adresse e-mail et son mot de passe. Les champs obligatoires sont indiqués, et lorsqu'ils sont remplis, les données sont envoyées au script "signup.php" pour traitement ultérieur, comme l'insertion dans une base de données ou la vérification des informations.
- Les champs obligatoires sont marqués avec l'attribut required pour s'assurer que l'utilisateur les remplit.

```
signup.php  creation.html X  personne.php  form
html > body > div.container > form
  <ul class="submenu">
  </ul>
</li>
</ul>
</nav>
</div>

class="container">
form action="signup.php" method="post">
  <h2>Créer un compte</h2>
  <label for="nom">Nom :</label>
  <input type="text" id="nom" name="nom" required>

  <label for="prenom">Prénom :</label>
  <input type="text" id="prenom" name="prenom" required>

  <label for="email">Adresse Email :</label>
  <input type="email" id="email" name="email" required>

  <label for="mdp">Mot de passe :</label>
  <input type="password" id="mdp" name="mdp" required>

  <button type="submit">Créer le compte</button>
form>
```

LOGIN

- **Chargement automatique de classes :**

La fonction **chargerClass** est enregistrée comme autoloader avec **spl_autoload_register**. Elle est utilisée pour charger les classes de manière dynamique à partir de fichiers PHP.

- **Initialisation de la session et de la connexion PDO :**

La session est démarrée avec **session_start()**. Ensuite, une connexion PDO à la base de données est établie en utilisant les informations de connexion fournies (localhost, root, "").

- **Traitement du formulaire :** Si le formulaire est soumis (vérification basée sur la présence de la clé **cree** dans **\$_POST**), les données de l'utilisateur (email et mot de passe) sont récupérées dans un tableau associatif **\$data**

```
require('connect.php');

function chargerClass($classname) {
    require $classname . '.php';
}

spl_autoload_register("chargerClass");
session_start();
$conn = new PDO("mysql:host=localhost;dbname=bd_personnel", 'root', '');
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
if (isset($_POST['cree'])) {
    $data = array(
        'email' => $_POST['email'],
        'password' => $_POST['password']
    );
}
```

```

}

// Création de l'objet User
$userManager = new User($conn);

// Exemple d'utilisation pour vérifier un utilisateur
$email = 'test@example.com';
$password = 'motdepasse';

$result = $userManager->verifUser($email, $password);
if ($result) {
    echo "Utilisateur trouvé.";
} else {
    echo "Utilisateur non trouvé.";
}

// Exemple d'utilisation pour insérer un nouvel utilisateur
$newEmail = 'nouvel_utilisateur@example.com';
$newPassword = 'nouveaumotdepasse';

$userManager->insertUser($newEmail, $newPassword);

?>

```

CLASSE USER

- L'objectif est d'instancier la classe User en lui passant la connexion à la base de données comme paramètre. Cette classe User est conçue pour gérer les opérations relatives aux utilisateurs dans la base de données.
- Pour vérifier un utilisateur, on appelle la méthode verifUser de l'objet UserManager en utilisant l'email et le mot de passe de l'utilisateur. En fonction du résultat de la vérification, un message approprié est affiché.
- Pour insérer un nouvel utilisateur, on appelle la méthode insertUser de l'objet UserManager en fournissant l'email et le mot de passe de l'utilisateur.

FONCTION D'AUTOLOAD ET SESSION

- **Autoload PHP:** Une fonction `chargerClass()` est définie pour charger les classes PHP. Ensuite, `spl_autoload_register()` est utilisée pour enregistrer cette fonction.
- **Session PHP:** La session est démarrée avec `session_start()` pour gérer les données de session.

```
spl_autoload_register("chargerClass");

session_start();

$conn = new PDO("mysql:host=localhost;dbname=mondb");
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

if (isset($_POST['cree'])) {
```

DÉFINITION DES CONSTANTES

- **SERVERNAME** : Nom du serveur de base de données (localhost dans cet exemple).
- **USERNAME** : Nom d'utilisateur de la base de données (root dans cet exemple).
- **PASSWORD** : Mot de passe de la base de données (vide dans cet exemple).
- **DATABASE** : Nom de la base de données à laquelle se connecter (bd_personnel dans cet exemple).

```
<?php
// Définition des constantes pour les
define('SERVERNAME', 'localhost');
define('USERNAME', 'root');
define('PASSWORD', '');
define('DATABASE', 'bd_personnel');

// Connexion à la base de données
```


EXCEPTION

- Utilisation d'un bloc try-catch pour gérer les erreurs potentielles lors de la connexion.
- Utilisation de la classe PDO pour établir la connexion en spécifiant le serveur, la base de données, le nom d'utilisateur et le mot de passe.
- Attribution de l'attribut PDO::ATTR_ERRMODE à PDO::ERRMODE_EXCEPTION pour activer la gestion des erreurs sous forme d'exceptions.
- Affichage d'un message de succès ou d'échec de la connexion en fonction du résultat du bloc try-catch.

```
// Connexion à la base de données
try {
    $conn = new PDO ("mysql:host=".SERVERM
    // Définition du mode de gestion des e
    $conn->setAttribute(PDO::ATTR_ERRMODE,
    echo "Connexion réussie";
} catch(PDOException $e) {
    echo "Échec de la connexion : " . $e->
}
?>
```

CLASSE PERSONNE

- **name:** nom de la personne
- **profession:** Profession de la personne.
- **address:** Adresse de la personne.
phone: Numéro de téléphone de la personne.
- email: Adresse e-mail de la personne.
- **datenais:** Date de naissance de la personne.
- **skills:** Compétences de la personne.
- company: Entreprise actuelle de la personne.
- **position:** Poste actuel de la personne.
- **start_date:** Date de début de l'emploi actuel.
- **end_date:** Date de fin de l'emploi actuel.
description: Description de l'emploi actuel.
- **school:** École ou université de la personne.
- **degree:** Diplôme obtenu par la personne.
- **start_year:** Année de début des études.
- **end_year:** Année de fin des études.
- **field_of_study:** Domaine d'étude de la personne

- **constructeur:**

Prend un tableau associatif de données sur la personne et appelle la méthode hydrate().

Méthode hydrate :

Remplit les propriétés de la classe avec les valeurs du tableau associatif.

- **Setters :**

Définit les valeurs des propriétés de la classe.

- **Getters :**

Récupère les valeurs des propriétés de la classe.

Classe Personnes

- **Propriétés :**

- **conn:** Objet de connexion à la base de données.

- **Constructeur :**

Prend un objet PDO en paramètre pour la connexion à la base de données.

```
class Personnes {  
    private $conn;  
  
    public function __construct(PDO $conn)  
    {  
        $this->conn = $conn;  
    }  
}
```

CONNEXION À LA BASE DE DONNÉES

```
spl_autoload_register("chargerClass");
session_start();

$conn = new PDO("mysql:host=localhost;dbname=bdpersonnel", 'root', 'root');
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

$stmt = $conn->prepare("SELECT * FROM personne LIMIT 1");
$stmt->execute();
$donnees = $stmt->fetch(PDO::FETCH_ASSOC);
?>

<div class="resume">
    <div class="resume_left">
        <div class="resume_content">
```

- **Connexion PDO:** Une connexion PDO est établie avec la base de données MySQL en utilisant les informations d'identification fournies.
- Récupération des Données de la Base de Données
- **Requête SQL:** Une requête SQL est préparée pour récupérer les données d'une table appelée "personne" en limitant le résultat à une seule entrée.
- **Exécution de la Requête:** La requête est exécutée avec `$stmt->execute()` et les données sont récupérées avec `$stmt->fetch(PDO::FETCH_ASSOC)` et stockées dans la variable `$donnees`.

MÉTHODE INSERER PERSONNE :

- Prépare une requête SQL pour insérer les données d'une personne dans la table personne.
- Lie les valeurs des propriétés de l'objet `Personne` des paramètres dans la requête SQL. Exécute la requête et affiche un message en cas de succès ou d'erreur.
- Cette structure de classes permet de créer des objets `Personne` avec des propriétés spécifiques et de les insérer dans une base de données en utilisant la classe `Personnes`. Les méthodes de ces classes permettent de manipuler facilement les données des personnes et de les stocker de manière organisée dans une base de données.

```
public function insererPersonne(Personne $personne) {  
    try {  
        $stmt = $this->conn->prepare("INSERT INTO personne (name, profession, address,  
  
        $stmt->bindValue(':name', $personne->getName());  
        $stmt->bindValue(':profession', $personne->getProfession());  
        $stmt->bindValue(':address', $personne->getAddress());  
        $stmt->bindValue(':phone', $personne->getPhone());  
        $stmt->bindValue(':email', $personne->getEmail());  
        $stmt->bindValue(':datenais', $personne->getDateNais());  
        $stmt->bindValue(':skills', $personne->getSkills());  
        $stmt->bindValue(':company', $personne->getCompany());  
        $stmt->bindValue(':position', $personne->getPosition());  
        $stmt->bindValue(':start_date', $personne->getStartDate());  
        $stmt->bindValue(':end_date', $personne->getEndDate());  
        $stmt->bindValue(':description', $personne->getDescription());  
        $stmt->bindValue(':school', $personne->getSchool());  
        $stmt->bindValue(':degree', $personne->getDegree());  
        $stmt->bindValue(':start_year', $personne->getStartYear());  
        $stmt->bindValue(':end_year', $personne->getEndYear());  
        $stmt->bindValue(':field_of_study', $personne->getFieldOfStudy());  
  
        $stmt->execute();  
    }  
}
```

CHARGEMENT DES DÉPENDANCES



















- **PHP Require:** Le code commence par l'utilisation de la fonction `require()` pour inclure le fichier de connexion à la base de données et initialise une variable `$row` pour stocker les données récupérées.

```
<?php
```

```
require('connect.php');  
$row = $stmt->fetch(PDO::FETCH_ASSOC);  
function chargerClass($classname) {  
    require $classname . '.php';  
}
```

The PHP logo, consisting of a dark blue square with a lighter blue square inside it, and the letters 'PHP' in white.

INTERFACE DE BASE DE DONNÉES

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	name	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
2	profession	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
3	address	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
4	phone	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
5	email	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
6	datenais	int			Non	Aucun(e)			 Modifier 
7	skills	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
8	company	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 
9	postion	varchar(30)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier 

THANK YOU FOR VISITING OUR SITE

OUR LOGO



SOURCES DE PROJET :
GITHUB
CHATGBT