**MARTY Dylan**

**Projet MyCoach**

**Page index**

La page est divisée en plusieurs sections définies par les balises <section> et des éléments HTML appropriés. Voici une vue d'ensemble des sections principales :

1. **Header (Barre de Navigation)**
   * La section du haut de la page contenant la navigation.
   * Les liens pointent vers différentes sections de la page.
2. **Accueil (Section #hero)**
   * Présente un héros avec un titre, une description et une liste de services.
   * Utilise également PHP pour extraire et afficher les sports proposés depuis une base de données.
3. **Coach (Section #tarifs)**
   * Présente des informations sur le coach sportif, y compris son nom, image, âge, sexe, contacts, et profil professionnel.
4. **Horaires (Section #horaires)**
   * Propose un formulaire de connexion avec des champs pour le login et le mot de passe.
   * Le formulaire envoie les données vers un script PHP pour l'authentification.
   * Il y a également un lien pour renvoyer vers une page d’inscription
5. **Footer (Pied de Page)**
   * Affiche le copyright et les droits d'auteur.

**Styles CSS**

Le fichier CSS "index2.css" définit les styles de la page pour une présentation cohérente et attrayante. Voici quelques-uns des sélecteurs CSS principaux utilisés dans le fichier :

* Styles pour le header (barre de navigation).
* Styles pour la section d'accueil, y compris les titres, paragraphes et boutons.
* Styles pour la section tarifs, y compris les informations sur le coach sportif.
* Styles pour la section horaires, y compris le formulaire de connexion.
* Styles pour le footer.

**Page index une fois connecté**

La page est divisée en plusieurs sections définies par les balises <section> et des éléments HTML appropriés. Voici une vue d'ensemble des sections principales :

1. **Header (Barre de Navigation)**
   * La section du haut de la page contenant la navigation.
   * Les liens pointent vers différentes sections de la page.
   * Il y a un bouton pour ce déconnecter et nous renvoyer vers la page index
2. **Accueil (Section #hero)**
   * Présente un héros avec un titre, une description et une liste de services.
   * Utilise également PHP pour extraire et afficher les sports proposés depuis une base de données.
3. **Coach (Section #tarifs)**
   * Présente des informations sur le coach sportif, y compris son nom, image, âge, sexe, contacts, et profil professionnel.
4. **Horaires (Section #horaires)**
   * Affiche une table de séances avec des informations sur les jours, les sports, les niveaux, les heures de début et de fin, les adresses et les villes.
   * Les données sont extraites de la base de données à l'aide de PHP.
5. **Footer (Pied de Page)**
   * Affiche le copyright et les droits d'auteur.

**Styles CSS**

Le fichier CSS "index2.css" définit les styles de la page pour une présentation cohérente et attrayante. Voici quelques-uns des sélecteurs CSS principaux utilisés dans le fichier :

* Styles pour le header (barre de navigation).
* Styles pour la section d'accueil, y compris les titres, paragraphes et boutons.
* Styles pour la section tarifs, y compris les informations sur le coach sportif.
* Styles pour la section horaires, y compris le formulaire de connexion.
* Styles pour le footer.

**Page d’inscription**

La page est composée de plusieurs sections :

1. **Header (Barre de Navigation)**
   * La barre de navigation contient un seul lien qui permet aux utilisateurs de retourner à la page d'accueil (index.php).
2. **Section Inscription (Section #inscription)**
   * Cette section contient le formulaire d'inscription.
   * Elle inclut un titre "Formulaire de connexion".
   * Le formulaire comporte des champs pour le login et le mot de passe.
   * En cas d'erreur lors de la soumission du formulaire, un message d'erreur est affiché.
3. **Footer (Pied de Page)**
   * Le pied de page contient le copyright et les droits d'auteur.

**Styles CSS**

Les styles CSS sont définis dans une feuille de style externe "index3.css". Le fichier CSS externe contient des règles de style pour la mise en page, les couleurs, les polices, etc.

**Script PHP**

La page utilise un script PHP pour gérer l'inscription. En cas d'erreur lors de la soumission du formulaire, le script renvoie un message d'erreur si l’utilisateur existe déjà, qui est ensuite affiché dans la section "Inscription".

# Script PHP "scriptInscription.php"

## Objectif du Script

Le script PHP "scriptInscription.php" a pour objectif de gérer le processus d'inscription d'un utilisateur. Il vérifie si le login saisi dans le formulaire d'inscription est unique dans la base de données et, le cas échéant, ajoute l'utilisateur à la base de données. En cas de non-uniqueness du login, il redirige l'utilisateur vers la page d'inscription avec un message d'erreur. Sinon, il redirige l'utilisateur vers la page d'accueil (ou toute autre page appropriée) après une inscription réussie.

## Fonctionnement du Script

Le script commence par initialiser un tampon de sortie (output buffer) en utilisant ob\_start(). Cela permet de stocker la sortie générée par le script au lieu de l'envoyer immédiatement au navigateur.

Le script inclut le fichier de connexion à la base de données en utilisant require("../required/connect.php"). Cette étape est essentielle pour établir une connexion à la base de données.

Il récupère les données saisies dans le formulaire d'inscription en utilisant $\_POST et les stocke dans des variables :

* + $nomSaisi : Le login saisi.
  + $motPasseSaisi : Le mot de passe saisi.

Le script exécute une requête SQL pour vérifier si le login saisi existe déjà dans la base de données :



Il utilise une clause SELECT pour rechercher un enregistrement ayant le même login que celui saisi.

Il exécute la requête SQL en utilisant $connexion->query($sql) et récupère la première ligne de résultat avec $result->fetch().

Si le login existe déjà dans la base de données (c'est-à-dire si $ligne n'est pas vide), le script effectue les actions suivantes :

* Redirige l'utilisateur vers la page d'inscription avec un message d'erreur indiquant que le login est déjà utilisé.
* Le script se termine avec exit() pour éviter l'exécution ultérieure du code.

 Si le login est unique (c'est-à-dire que $ligne est vide), le script exécute une requête d'insertion pour ajouter l'utilisateur à la base de données :

$sql = "INSERT INTO `utilisateur` (`id`, `login`, `mdp`) VALUES (NULL, '$nomSaisi', '$motPasseSaisi');";

Il insère le nouveau login et le mot de passe dans la table "utilisateur".

Après une inscription réussie, le script redirige l'utilisateur vers la page d'accueil (ou toute autre page appropriée) en utilisant header("location:../index.php").

## Fermeture de la Connexion à la Base de Donnée avec exit().

# Script PHP "scriptIdentification.php"

## Objectif du Script

Le script PHP "scriptIdentification.php" a pour objectif de gérer le processus d'identification de l'utilisateur. Il vérifie si le nom d'utilisateur (login) et le mot de passe saisis correspondent aux enregistrements dans la base de données. Si les informations sont correctes, l'utilisateur est redirigé vers la page de consultation des frais. Sinon, un message d'erreur est affiché, et l'utilisateur est redirigé vers la page d'accueil.

## Fonctionnement du Script

1. Le script commence par initialiser un tampon de sortie (output buffer) en utilisant ob\_start(). Cela permet de stocker la sortie générée par le script au lieu de l'envoyer immédiatement au navigateur.
2. Le script inclut le fichier de connexion à la base de données en utilisant require("../required/connect.php"). Cette étape est essentielle pour établir une connexion à la base de données.
3. Il récupère les données saisies dans le formulaire d'identification en utilisant $\_POST et les stocke dans des variables :
   * $nomSaisi : Le login saisi.
   * $motPasseSaisi : Le mot de passe saisi.
4. Le script exécute une requête SQL pour récupérer le mot de passe enregistré dans la base de données qui correspond au login saisi :



Il utilise une clause SELECT pour récupérer le mot de passe associé au login saisi.

 Il exécute la requête SQL en utilisant $connexion->query($sql) et récupère la première ligne de résultat avec $result->fetch().

 Le mot de passe récupéré de la base de données est stocké dans la variable $motPasseBdd.

 Le script compare le mot de passe saisi avec le mot de passe de la base de données pour vérifier l'authenticité de l'utilisateur :

* Si le mot de passe saisi correspond à celui enregistré dans la base de données, l'utilisateur est considéré comme authentifié.
* Si les mots de passe ne correspondent pas, l'utilisateur reçoit un message d'erreur et est redirigé vers la page d'accueil.

 Si l'authentification est réussie, le script exécute une autre requête SQL pour récupérer l'ID de l'utilisateur correspondant au login :

php



1. Il exécute cette requête SQL, récupère l'ID de l'utilisateur, démarre une session avec session\_start(), puis stocke l'ID de l'utilisateur dans $\_SESSION['user\_id'] et le nom de l'utilisateur dans $\_SESSION['user\_name'].
2. Enfin, l'utilisateur est redirigé vers la page de consultation des frais (ou toute autre page appropriée) avec header("location:../index\_connecter.php").
3. Si l'authentification échoue (mot de passe incorrect ou login incorrect), un message d'erreur est affiché avec echo, et l'utilisateur est redirigé vers la page d'accueil avec header("location:../index.php#horaires").
4. Le script termine son exécution en fermant la connexion à la base de données en utilisant $connexion = null.

# Connexion à une Base de Données MySQL avec PDO

## Objectif du Code

Le code PHP que vous avez fourni a pour objectif d'établir une connexion à une base de données MySQL à l'aide de PDO (PHP Data Objects). Il définit des constantes pour les paramètres de connexion, puis essaie de se connecter à la base de données. En cas de succès, la connexion est établie et peut être utilisée pour interagir avec la base de données.

## Fonctionnement du Code

1. **Définition des Constantes de Connexion** : Le code commence par définir des constantes pour les paramètres de connexion à la base de données. Ces constantes stockent les informations suivantes :
   * SERVEUR : L'adresse du serveur MySQL. Dans cet exemple, c'est "localhost", ce qui signifie que la base de données est sur la même machine que le serveur web. Vous pouvez modifier cette constante pour pointer vers un serveur distant si nécessaire.
   * BASE : Le nom de la base de données à laquelle vous souhaitez vous connecter.
   * NOM : Le nom d'utilisateur MySQL pour la connexion.
   * MOTPASSE : Le mot de passe associé à l'utilisateur MySQL.
2. **Connexion à la Base de Données** : Le code utilise un bloc try...catch pour gérer les exceptions qui peuvent survenir lors de la tentative de connexion à la base de données. Voici comment cela fonctionne :
   * La classe PDO est utilisée pour établir la connexion à la base de données. Le constructeur de PDO prend trois arguments :
     + "mysql:host=".SERVEUR : C'est la chaîne de connexion qui spécifie le type de base de données (MySQL) et l'adresse du serveur.
     + BASE : Le nom de la base de données à laquelle vous souhaitez vous connecter.
     + NOM : Le nom d'utilisateur MySQL.
     + MOTPASSE : Le mot de passe associé à l'utilisateur MySQL.
3. **Gestion des Exceptions** : Si la tentative de connexion échoue, une exception de type Exception est générée. Le bloc catch capture cette exception et affiche un message d'erreur à l'utilisateur en utilisant die. Ce message d'erreur peut être personnalisé en utilisant $e->getMessage() pour obtenir des informations spécifiques sur l'erreur.
4. Une fois que la connexion est établie avec succès, la variable $connexion est prête à être utilisée pour effectuer des opérations de base de données, telles que l'exécution de requêtes SQL.

# Script PHP pour Déconnexion

## Objectif du Code

Le code PHP a pour objectif de déconnecter un utilisateur en supprimant sa session. Il effectue plusieurs opérations pour garantir que l'utilisateur est correctement déconnecté.

## Fonctionnement du Code

1. **Démarrage de la Session** : Le code commence par démarrer la session en utilisant session\_start(). Cela est nécessaire pour accéder aux variables de session et gérer la session de l'utilisateur.
2. **Nettoyage de la Session** : Le code effectue plusieurs opérations pour nettoyer la session de l'utilisateur :
   * session\_unset(): Cette fonction supprime toutes les variables de session enregistrées. Cela garantit que toutes les données de session sont effacées.
   * session\_destroy(): Cette fonction détruit la session en cours. Cela signifie que la session actuelle est fermée et toutes ses données sont supprimées.
3. **Fermeture de la Session** : Le code utilise session\_write\_close() pour s'assurer que la session est correctement fermée. Cela permet d'éviter les problèmes potentiels liés à la réouverture de la session.
4. **Suppression du Cookie de Session** : Le code utilise setcookie(session\_name(), '', 0, '/') pour supprimer le cookie de session. Cela signifie que le navigateur du client ne stockera plus le cookie de session, ce qui empêche toute session ultérieure avec le même ID de session.
5. **Régénération de l'ID de Session** : Le code utilise session\_regenerate\_id(true) pour régénérer l'ID de session. Cela ajoute une couche de sécurité en générant un nouvel ID de session pour l'utilisateur. Cela rend plus difficile l'attaque de fixation de session.
6. **Redirection vers la Page d'Accueil** : Enfin, le code utilise header("location:../index.php#horaires") pour rediriger l'utilisateur vers la page d'accueil (ou une autre page appropriée) après la déconnexion.

# Gestion des Session Utilisateurs(vérification de si l’utilisateur est connecter)

## Objectif du Code

Le code PHP a pour objectif de gérer la session de l'utilisateur, notamment en vérifiant si un utilisateur est authentifié et en protégeant l'accès à certaines pages en fonction de la session. Il effectue des vérifications pour s'assurer que l'utilisateur est connecté et qu'il a le droit d'accéder à certaines parties du site.

## Fonctionnement du Code

1. **Démarrage de la Session** : Le code commence par démarrer une session en utilisant session\_start(). Cela permet d'accéder aux variables de session et de gérer la session de l'utilisateur.
2. **Vérification de la Session** : Le code effectue les étapes suivantes pour vérifier la session de l'utilisateur :
   * Il vérifie si la variable de session user\_name existe. Si elle existe, elle est stockée dans la variable $nom. Sinon, l'utilisateur est redirigé vers la page d'accueil (index.php#horaires).
   * Il vérifie également si la variable de session user\_id existe. Si elle existe, elle est stockée dans la variable $id. Sinon, l'utilisateur est également redirigé vers la page d'accueil.
3. **Vérification de l'Authentification** : Le code effectue une vérification supplémentaire en comparant l'ID de session ($id) avec l'ID de l'utilisateur stocké dans la base de données. Voici comment cela fonctionne :
   * Il exécute une requête SQL pour sélectionner l'ID de l'utilisateur à partir de la base de données en utilisant le nom d'utilisateur ($nom) stocké dans la variable de session.
   * Si l'ID de session ($id) n'est pas égal à l'ID de l'utilisateur dans la base de données, l'utilisateur est redirigé vers la page d'accueil. Cela garantit que l'utilisateur authentifié ne peut pas accéder à des ressources protégées par l'authentification.
4. **Protection des Pages** : Ce code peut être placé au début des pages que vous souhaitez protéger. Si un utilisateur n'est pas connecté ou s'il tente d'accéder à une page sans autorisation, il sera automatiquement redirigé vers la page d'accueil.

Bas du formulaire