Formulaires

I. Utilisation des inclusions include/require

```
<?php require_once(__DIR__ . 'header.php'); ?>
```

Les instructions include et require sont des fonctionnalités fondamentales en PHP qui permettent d'inclure et d'exécuter le contenu d'un fichier PHP dans un autre. Elles sont essentielles pour organiser et réutiliser le code de manière efficace, en favorisant la modularité et la maintenabilité des projets PHP.

- **include** : Permet d'inclure un fichier dans un script PHP. Si le fichier inclus n'est pas trouvé, un avertissement est émis mais le script continue son exécution.

- **require** : Fonctionne de manière similaire à include, mais génère une erreur fatale si le fichier inclus n'est pas trouvé, interrompant ainsi l'exécution du script.

- **require_once** : Inclut un fichier PHP une seule fois pendant l'exécution du script. Si le fichier a déjà été inclus, il ne sera pas inclus à nouveau.

Avantages de l'inclusion de fichiers :

- Réutilisation du code : Les fonctions, les classes et d'autres éléments peuvent être définis dans un fichier et inclus dans plusieurs scripts PHP, évitant ainsi la duplication de code.

- Organisation : Permet de diviser logiquement le code en fichiers distincts selon leur fonctionnalité, facilitant ainsi la compréhension et la maintenance du code.

- Modularité : Facilite l'ajout, la modification ou la suppression de fonctionnalités en manipulant simplement les fichiers inclus sans avoir à modifier tous les scripts.

Exemple d'utilisation :

Supposons que nous ayons un fichier "fonctions.php" contenant des fonctions utilitaires, telles que des fonctions mathématiques. Nous pouvons inclure ce fichier dans d'autres scripts PHP en utilisant les instructions include ou require, comme suit :

```php
// Utilisation de require
require 'fonctions.php';

// Appel des fonctions incluses
$resultat = addition(5, 3);
echo "Résultat de l'addition : " . $resultat;
```

Gestion des erreurs :

Il est important de prendre en compte la gestion des erreurs lors de l'utilisation des instructions include et require. En cas d'échec de l'inclusion d'un fichier, il est nécessaire de gérer les avertissements ou les erreurs générés pour assurer le bon fonctionnement de l'application.

Exercice : Création d'une page web avec structure header, footer et menu de navigation

- Créez un nouveau projet PHP sur votre serveur local.
- Organisez votre projet en utilisant une structure de dossier comprenant au moins le répertoire suivant : "includes".
- Dans le répertoire "includes", créez deux fichiers PHP : "header.php" et "footer.php".
- Dans le fichier "header.php", créez une structure de header contenant un menu de navigation avec des liens vers différentes pages de votre site.
- Dans le fichier "footer.php", créez une structure de footer basique, par exemple avec des informations de copyright.
- Dans le répertoire de base créez au moins deux fichiers PHP représentant différentes pages de votre site, par exemple "index.php" et "about.php".
- Utilisez une instruction d'inclusion pour inclure le fichier "header.php" au début de chaque page, et le fichier "footer.php" à la fin de chaque page.
- Testez votre application en accédant aux différentes pages et vérifiez que la structure du header et du footer est correctement affichée sur chaque page.

II. Manipulation des données d'un formulaire HTML avec PHP

- **Paramètres :** page.php?param1=valeur1&param2=valeur2&param3=valeur3
- **$_GET** : Contient les données envoyées à votre script via la méthode GET. Les données sont généralement incluses dans l'URL. On peut s'amuser à changer le paramètre dans l'URL. La seule limite est la longueur de l'URL (en général, pas plus de 256 caractères).
- **$_POST** : Contient les données envoyées à votre script via la méthode POST. Les données sont généralement incluses dans le corps de la requête HTTP. l'utilisateur ne les verra donc pas passer dans la barre d'adresse.
- **if (!isset($_SESSION['LOGGED_USER']))** pour tester si la variable existe

- **unset($viariable)** = détruit la variable
- **strip_tags()** = supprime les balises HTML et PHP d'une chaîne de caractères.
- **filter_var($getData['email'], FILTER_VALIDATE_EMAIL)** pour vérifier que l'email est valide
- **empty($getData['message'])** pour vérifier que ce n'est pas vide

III. Sécurisation des données et validation des formulaires avec PHP

ATTENTION SI ON MET CE CODE : *** Message : <script>alert('Badaboum')</script> ***

Pour ignorer le code HTML, il suffit d'utiliser la fonction **htmlspecialchars** pour éviter la faille XSS. La faille XSS (pour cross-site scripting) est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages, pour le faire exécuter à vos visiteurs.

*** Message : <?php echo htmlspecialchars($_POST['message']); ?> ***

Champs cachés quand même visibles dans le code source

Les variables superglobales : ce sont des variables prédéfinies qui sont toujours accessible, quel que soit le contexte du script et donc disponibles dans toutes les portées (locales, globales, etc.) et utilisables n'importe où dans le script. Voici quelques exemples :

- **$_GET** : Contient les données envoyées à votre script via la méthode GET. Les données sont généralement incluses dans l'URL.
- **$_POST** : Contient les données envoyées à votre script via la méthode POST. Les données sont généralement incluses dans le corps de la requête HTTP.
- **$_COOKIE** : Contient les données des cookies envoyées par le client au serveur, qui sont stockées par votre site sur l'ordinateur du visiteur. Les cookies sont de petits fichiers que l'on peut écrire sur la machine du visiteur pour retenir par exemple son nom. On crée un cookie avec la fonction setcookie() .
- **$_REQUEST** : Contient à la fois les données de $_GET, $_POST et $_COOKIE. Il est souvent utilisé pour obtenir des données de l'utilisateur sans se soucier de la méthode d'envoi.
- **$_SESSION** : Contient les variables de session. Ces variables permettent de stocker des informations spécifiques à une session utilisateur, qui seront automatiquement transmises de page en page pendant toute la durée de visite d'un internaute sur votre site. Il faut au préalable activer les sessions en appelant la fonction session_start() .

- **$_SERVER** : Contient des informations sur le serveur et l'environnement d'exécution. Par exemple, $_SERVER['HTTP_USER_AGENT'] contient le User-Agent du navigateur du client.

- **$_FILES** : Contient des informations sur les fichiers téléchargés via un formulaire enctype="multipart/form-data".

- **$_ENV** : Contient les variables d'environnement du serveur.

- **$GLOBALS** : Contient toutes les variables globales.

Exercice : création d'un formulaire de contact avec validation et affichage des données soumises

- Créez deux fichiers PHP sur votre serveur local, nommés "index.php" et "submit_contact.php".

- Dans le fichier "index.php", à partir de la structure fournie, créez dans la balise de classe « container » la structure HTML d'un formulaire de contact avec les champs suivants : "Email" (type email) / "Message" (type textarea) / Un bouton "Envoyer" pour soumettre le formulaire.

- Définissez l'attribut "action" du formulaire pour qu'il envoie les données à "submit_contact.php" en utilisant la méthode GET.

- Dans le fichier "submit_contact.php", récupérez les données soumises via la superglobale $_GET.

- Vérifiez si les données sont valides : l'email doit être au format valide et le champ "Message" ne doit pas être vide.

- Si les données ne sont pas valides, affichez un message d'erreur approprié.

- Si les données sont valides, affichez une nouvelle page HTML avec un message confirmant la réception du message, ainsi que les informations soumises (email et message).

- Testez votre formulaire en soumettant des données valides et invalides pour vous assurer que la validation fonctionne correctement.

Exercice : V2 en POST et V3

- Créez trois fichiers PHP sur votre serveur local, nommés "index.php", "login.php" et "submit_contact.php".

- Dans le fichier "index.php", intégrez le fichier "login.php" en utilisant l'instruction require_once.

- Dans le fichier "login.php", récupérez les données soumises via la superglobale $_POST.

- Validez les données soumises : vérifiez si l'email est au format valide et si le mot de passe est correct.

- Si les données ne sont pas valides ou si l'utilisateur n'est pas identifié, affichez le formulaire de connexion avec un message d'erreur le cas échéant.

- Si l'utilisateur est identifié avec succès, affichez un message de bienvenue.

- Dans le fichier "submit_contact.php", récupérez les données soumises via la superglobale $_POST.

- Validez les données soumises pour l'email et le message : vérifiez si l'email est au format valide et si le champ message n'est pas vide.

- Si les données ne sont pas valides, affichez un message d'erreur approprié.

- Si les données sont valides, affichez un message de confirmation de la réception du message, ainsi que les informations soumises (email et message).

- Testez votre formulaire en soumettant des données valides et invalides pour vous assurer que la validation fonctionne correctement.

IV. Les sessions

Les sessions en PHP sont un mécanisme permettant de stocker des données de manière temporaire et persistante tout au long de la navigation d'un utilisateur sur un site web. Les sessions sont particulièrement utiles pour conserver l'état de l'utilisateur entre différentes pages ou requêtes HTTP.

1. Démarrer une session :

La première étape pour utiliser les sessions en PHP est de démarrer la session en appelant la fonction session_start(). Cette fonction doit être appelée avant tout autre sortie HTTP, y compris les balises HTML. Voici comment démarrer une session : **session_start();**

2. Stocker des données dans la session :

Une fois la session démarrée, vous pouvez stocker des données dans la session en utilisant la superglobale $_SESSION. Les données stockées dans $_SESSION seront disponibles tant que la session est active. Voici un exemple de stockage de données dans une session :

```php
<?php
session_start();

// Stocker des données dans la session
$_SESSION['username'] = 'john_doe';
$_SESSION['user_role'] = 'admin';
?>
```

```php
<?php
session_start();

// Accéder aux données de la session
echo $_SESSION['username']; // Affiche "john_doe"
echo $_SESSION['user_role']; // Affiche "admin"
?>
```

3. Accéder aux données de la session :

Pour accéder aux données stockées dans la session, vous pouvez simplement utiliser la superglobale $_SESSION. Voici comment accéder aux données de la session :

4. Effacer les données de la session :

Pour effacer des données spécifiques de la session, vous pouvez utiliser la fonction unset() en passant la clé des données à effacer. Par exemple : **unset($_SESSION['username']);**

5. Détruire une session :

Pour détruire complètement une session, vous pouvez utiliser la fonction session_destroy(). Cela supprimera toutes les données de la session en cours et mettra fin à la session. Voici comment détruire une session : **session_destroy();**

6. Exercice

- Créez trois fichiers PHP sur votre serveur local : "index.php", "login.php" et "logout.php".
- Dans le fichier "index.php", créez une page d'accueil qui affiche un message de bienvenue si l'utilisateur est connecté, sinon affichez un lien de connexion.
- Dans le fichier "login.php", créez un formulaire de connexion demandant un nom d'utilisateur et un mot de passe. Vérifiez si les informations de connexion sont valides et démarrez une session si l'utilisateur est authentifié avec succès.
- Dans le fichier "logout.php", détruisez la session en cours lorsque l'utilisateur se déconnecte.
- Utilisez la fonction session_start() pour démarrer la session dans tous les fichiers où vous manipulez des sessions.
- Stockez les informations de l'utilisateur dans la session une fois qu'il est connecté avec succès.
- Affichez un message de bienvenue personnalisé sur la page d'accueil une fois que

l'utilisateur est connecté.

- Assurez-vous de sécuriser votre système de connexion en vérifiant les informations de connexion et en empêchant l'accès non autorisé aux pages protégées.

- Testez votre système de connexion en vous connectant avec des identifiants valides et en vous déconnectant avec succès.