Gestion des donnée d'un formulaire

Par Robert DIASSÉ

Introduction

La gestion des données en PHP est essentielle pour développer des applications web interactives. Les formulaires permettent de recueillir des informations auprès des utilisateurs et de les traiter sur le serveur. Ce cours couvrira les méthodes de transmission des données via les formulaires, l'utilisation de l'attribut `action`, la récupération des données, les traitements de base pour assurer la sécurité des données, et quelques exemples pratiques.

1. Méthodes de Transmission des Données

En PHP, les formulaires HTML peuvent envoyer des données au serveur en utilisant deux méthodes principales : `GET` et `POST`.

Méthode GET

- **Utilisation :** Envoie les données via l'URL.
- **Avantages :** Les données sont visibles dans l'URL, ce qui facilite le partage de liens. Utile pour les requêtes idempotentes (recherches).
- **Inconvénients :** Limitation de la taille des données (environ 2048 caractères). Moins sécurisé car les données sont visibles dans l'URL.
- **Exemple :**

```
<form method="GET" action="traitement.php">
    <input type="text" name="nom" placeholder="Votre nom">
    <input type="submit" value="Envoyer" name="submit">
</form>
```

Méthode POST

- **Utilisation :** Envoie les données dans le corps de la requête HTTP.
- **Avantages :** Pas de limitation significative de la taille des données. Plus sécurisé que GET car les données ne sont pas visibles dans l'URL.

- **Inconvénients :** Les données ne sont pas visibles dans l'URL, ce qui rend le partage de liens direct impossible.
- **Exemple :**

```
<form method="POST" action="traitement.php">
    <input type="text" name="nom" placeholder="Votre nom">
    <input type="submit" value="Envoyer" name="submit">
</form>
```

---

2. L'attribut `action`

L'attribut `action` dans un formulaire HTML spécifie l'URL de la page qui traitera les données soumises par le formulaire. Si l'attribut `action` est omis, les données seront envoyées à la même page.

- **Exemple avec action:**

```
<form method="POST" action="traitement.php">
    <input type="text" name="nom" placeholder="Votre nom">
    <input type="submit" value="Envoyer" name="submit">
</form>
```

- **Exemple sans action (soumission à la même page) :**

```
<form method="POST">
    <input type="text" name="nom" placeholder="Votre nom">
    <input type="submit" value="Envoyer" name="submit">
</form>
```

3. Récupération des Données avec PHP

Les données envoyées par un formulaire HTML peuvent être récupérées en PHP à l'aide des variables superglobales `$_GET` et `$_POST`, qui sont des tableaux associatifs contenant les données soumises.

Variable Superglobale `$_GET`

- Contient les données envoyées par la méthode GET.
- Exemple de récupération de données avec GET:

```
if (isset($_GET['submit'])) {
    $nom = $_GET['nom'];
    echo "Nom (GET) : " . htmlspecialchars($nom);
}
```

Variable Superglobale `$_POST`

- Contient les données envoyées par la méthode POST.
- Exemple de récupération de données avec POST:

```
if (isset($_POST['submit'])) {
    $nom = $_POST['nom'];
    echo "Nom (POST) : " . htmlspecialchars($nom);
}
```

4. Vérification du Bouton de Soumission

Pour vérifier si le bouton de soumission a été appuyé, vous pouvez utiliser `isset()` sur l'élément correspondant du tableau associatif `$_GET` ou `$_POST`.

- **Exemple :**

```
if (isset($_POST['submit'])) {
    // Traitement des données
}
```

5. Sécurisation de Base des Données

Lorsque vous travaillez avec des formulaires et des données utilisateur, il est crucial de sécuriser les données pour éviter les failles de sécurité courantes comme les injections SQL, les scripts intersites (XSS), etc.

Validation et Nettoyage des Données

- **Validation :** Vérifiez que les données reçues correspondent aux attentes (par exemple, vérifier qu'une adresse email a un format valide).
- **Nettoyage :** Éliminez les caractères indésirables ou dangereux des données reçues.

Fonctions Utiles en PHP

- `isset()`: Vérifie si une variable est définie.
- `empty()`: Vérifie si une variable est vide.
- `trim()`: Supprime les espaces en début et fin de chaîne.
- `htmlspecialchars()`: Convertit les caractères spéciaux en entités HTML pour prévenir les XSS.
- `filter_var()`: Valide et filtre les données selon des options spécifiques.

Exemple de Traitement Sécurisé des Données

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['submit'])) {
        if (isset($_POST['nom']) && !empty($_POST['nom'])) {
            $nom = trim($_POST['nom']);
```

```
            $nom = htmlspecialchars($nom);
            echo "Nom nettoyé et sécurisé : " . $nom;
        } else {
            echo "Veuillez entrer un nom.";
        }
    }
}
```

6. Exemples Pratiques

Exemple 1 : Formulaire de Contact

```
<form method="POST" action="contact.php">
    <input type="text" name="nom" placeholder="Votre nom" required>
    <input type="email" name="email" placeholder="Votre email" required>
    <textarea name="message" placeholder="Votre message" required></textarea>
    <input type="submit" value="Envoyer" name="submit">
</form>
```

```
// contact.php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['submit'])) {
        $nom = trim($_POST['nom']);
        $nom = htmlspecialchars($nom);

        $email = trim($_POST['email']);
        $email = htmlspecialchars($email);
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            echo "Adresse email invalide.";
            exit;
        }

        $message = trim($_POST['message']);
        $message = htmlspecialchars($message);

        // Enregistrer ou envoyer les données
        echo "Merci, $nom. Votre message a été envoyé.";
    }
}
```

Exemple 2 : Formulaire de Connexion

```
<form method="POST" action="login.php">
    <input type="text" name="username" placeholder="Nom d'utilisateur" required>
    <input type="password" name="password" placeholder="Mot de passe" required>
```

```
    <input type="submit" value="Connexion" name="submit">
</form>
```

```php
// login.php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['submit'])) {
        $username = trim($_POST['username']);
        $username = htmlspecialchars($username);

        $password = trim($_POST['password']);
        $password = htmlspecialchars($password);

        // Validation et vérification des données
        if (!empty($username) && !empty($password)) {
            // Simuler une vérification de mot de passe
            if ($username == "admin" && $password == "password") {
                echo "Connexion réussie. Bienvenue, $username.";
            } else {
                echo "Nom d'utilisateur ou mot de passe incorrect.";
            }
        } else {
            echo "Veuillez remplir tous les champs.";
        }
    }
}
```

Conclusion

Il est primordial de s'aider de la documentation officielle PHP pour évoluer et approfondir ses connaissances. La documentation est une ressource inestimable pour comprendre les fonctionnalités et les meilleures pratiques de PHP. Vous pouvez la trouver ici : Documentation PHP.

Exercices

1. Créez un formulaire de saisie de produit (nom, prix, quantité) et affichez les données saisies.
2. Créez un formulaire de connexion avec vérification du nom d'utilisateur et du mot de passe.
3. Créez un formulaire de contact et envoyez les données par email en utilisant la fonction `mail()` de PHP.