Série d'exercices sur les Formulaires en PHP

Centre BTS ALKENDI Casablanca

Dr. Ayman NAIM

March 4, 2025

Contents

Introduction

Cette série d'exercices couvre les notions fondamentales et avancées sur la création et la gestion de formulaires en PHP. Vous apprendrez à construire des formulaires HTML, à récupérer et valider les données côté serveur, ainsi qu'à sécuriser et améliorer l'expérience utilisateur grâce à des pratiques de développement modernes.

NIVEAU DÉBUTANT

Exercice 1 : Formulaire simple et récupération des données

Mise en situation :
Créez un formulaire HTML avec un champ `input` pour le nom de l'utilisateur et un bouton `submit`. En PHP, récupérez et affichez le nom saisi.

Objectif :

- Comprendre la syntaxe d'un formulaire HTML basique.
- Utiliser `$_POST` pour récupérer la valeur du champ.
- Afficher la valeur saisie à l'écran.

Indications techniques :

- Utiliser les attributs `method="POST"` et `action="traitement.php"`.
- Sanitiser la sortie avec `htmlspecialchars()` pour éviter les injections XSS.

Contraintes et défis :

- Assurer que la récupération de données se fait avant tout affichage (pas d'erreur de variable non définie).

Exercice 2 : Formulaire avec différents types de champs

Mise en situation :
Réalisez un formulaire contenant un `input type="text"`, un `input type="email"`, un `select` et un `textarea`. À la soumission, affichez les données saisies.

Objectif :

- Manipuler différents types de champs HTML.
- Récupérer leurs valeurs côté serveur avec `$_POST`.

Indications techniques :

- Ne pas oublier d'ajouter l'attribut `name` à chaque champ.
- Vérifier la structure HTML pour une bonne mise en forme.

Contraintes et défis :

- Gérer correctement le `select` pour afficher la valeur choisie.

Exercice 3 : Validation basique de champs obligatoires

Mise en situation :
Ajoutez une validation pour vérifier que tous les champs sont remplis. Si un champ est vide, affichez un message d'erreur.

Objectif :

- Vérifier la présence de données avec `empty()`.
- Afficher un message d'erreur au lieu des résultats si un champ est vide.

Indications techniques :

- Renvoyer l'utilisateur au formulaire ou afficher un message d'erreur approprié.

Contraintes et défis :

- Ajouter de la lisibilité (ex. colorer les messages d'erreur en rouge).

NIVEAU INTERMÉDIAIRE

Exercice 4 : Validation avancée (email, longueur, etc.)

Mise en situation :
À partir du formulaire précédent, validez le champ email pour vérifier son format. Ajoutez une contrainte de longueur minimale pour un champ texte (ex. nom).

Objectif :

- Utiliser `filter_var()` ou `filter_input()` pour valider l'email.
- Vérifier la longueur d'un champ (ex. `strlen($nom) > 3`).

Indications techniques :

- Tester les cas limites (email invalide, champ trop court).

Contraintes et défis :

- Gérer l'affichage de plusieurs erreurs simultanément (tableau `$errors`).

Exercice 5 : Nettoyage et protection XSS

Mise en situation :
Pour éviter toute injection de code HTML ou JavaScript, nettoyez systématiquement les données avec `htmlspecialchars()` ou `strip_tags()`.

Objectif :

- Protéger votre application contre les attaques XSS.

- Centraliser la fonction de nettoyage (ex. fonction `nettoyer(`*valeur*`)`).

Indications techniques :

- Appliquer la fonction de nettoyage aux champs avant l'affichage.

Contraintes et défis :

- Choisir la bonne combinaison entre `strip_tags()` et `htmlspecialchars()` pour conserver les sauts de ligne ou certaines balises autorisées.

Exercice 6 : Retour utilisateur et persistance des données dans le formulaire

Mise en situation :
En cas d'erreur de validation, le formulaire doit se réafficher avec les données précédemment saisies pour éviter à l'utilisateur de tout retaper.

Objectif :

- Renseigner les attributs `value=""` des champs `input` avec la dernière saisie.
- Préremplir la `textarea` et le `select`.

Indications techniques :

- Utiliser `echo isset($_POST['champ']) ? $_POST['champ'] : ''` pour recharger la valeur.

Contraintes et défis :

- Gérer le cas où la validation échoue partiellement (certains champs corrects, d'autres erronés).

NIVEAU AVANCÉ

Exercice 7 : Multiples étapes (Formulaire en plusieurs pages)

Mise en situation :
Créez un formulaire en deux étapes (Page 1 : informations personnelles, Page 2 : informations professionnelles). Vous devez transférer les données de la première page à la deuxième, puis afficher un récapitulatif complet.

Objectif :

- Gérer le passage des données d'une page à l'autre.
- Concaténer les informations et les afficher en fin de processus.

Indications techniques :

- Utiliser `session_start()` ou passer les données via des champs `hidden`.

Contraintes et défis :

- Conserver la validation pour chaque étape.

Exercice 8 : Utilisation de la méthode GET pour un moteur de recherche

Mise en situation :
Créez un mini moteur de recherche qui utilise `method="GET"` pour transmettre le mot-clé. Affichez ensuite un message du type : "Résultats pour la recherche : *mot-clé*".

Objectif :

- Comprendre les avantages de `GET` (paramètres visibles, partageables).
- Vérifier la présence d'un paramètre `query` dans l'URL.

Indications techniques :

- Utiliser `$_GET['query']` pour récupérer la valeur.

Contraintes et défis :

- Sanitiser la valeur pour éviter les problèmes de sécurité.

Exercice 9 : Téléchargement d'un fichier via formulaire

Mise en situation :
Mettez en place un formulaire permettant d'uploader une image. Le script doit vérifier le type et la taille du fichier avant de l'enregistrer dans un dossier `uploads/`.

Objectif :

- Utiliser `enctype="multipart/form-data"`.
- Manipuler `$_FILES` (nom, type, taille, tmp_name).

Indications techniques :

- Tester la limite de taille (ex. 2 Mo).
- Accepter uniquement les images (type MIME `image/jpeg`, `image/png`, etc.).

Contraintes et défis :

- Utiliser `move_uploaded_file()` pour déplacer le fichier de la zone temporaire.

Exercice 10 : Formulaire complet avec validation, messages d'erreur et upload

Mise en situation :
Combinez les approches précédentes pour créer un formulaire complexe :

- Champs texte et email.
- Zone de sélection.
- Champ de téléchargement de fichier.
- Validation et affichage des erreurs.

Objectif :

1. Construire un formulaire HTML riche.
2. Appliquer une validation complète (formats, longueur, fichiers).
3. Gérer les messages d'erreur et la persistance des données dans les champs.

Indications techniques :

- Utiliser un tableau `$errors` pour lister toutes les erreurs détectées.
- Se limiter aux images et vérifier la taille maximale de l'upload.

Contraintes et défis :

- Proposer une page de confirmation récapitulative lorsque tout est valide.

Conclusion

Ces exercices vous permettront de maîtriser les différents aspects de la gestion des formulaires en PHP, du simple formulaire de contact à la mise en place d'un formulaire complexe avec uploads, validation et messages d'erreur. La sécurisation des données, la gestion de la persistance et l'amélioration de l'expérience utilisateur sont au cœur de ces défis, vous assurant ainsi un socle solide pour toutes vos futures applications web.