Table des matières

[LA METHODE POST 2](#_Toc73783106)

[LES FORMULAIRES EN PHP 2](#_Toc73783107)

[Comment gérer un formulaire en PHP : 2](#_Toc73783108)

[La récupération des champs de formulaire en POST 4](#_Toc73783109)

# LA METHODE POST

La méthode POST comme la méthode GET sert à envoyer des données vers une page PHP.

Contrairement à la méthode GET, la méthode POST n’envoie pas les données en « clair » ce qui permet de cacher les données qui transit d’une page à une autre.

Contrairement à la méthode GET vu que les données sont cachées nous ne pouvons les envoyer simplement comme des données en GET par une URL.

Nous allons donc utiliser un formulaire pour nous servir de la méthode POST.

# LES FORMULAIRES EN PHP

Un Formulaire en PHP peut utiliser soit la méthode GET soit la POST, favorisez tout le temps la méthode POST pour éviter l’interception de données sensibles, comme un mot de passe.

## Comment gérer un formulaire en PHP :

Un simple formulaire en HTML sur un fichier index.php

L’attribut action de notre element form => Détermine l’endroit ou est envoyé notre formulaire

L’attribut method de notre element form => Détermine le type de method GET ou POST utilisé par notre formulaire

Voici à quoi ressemble l’url quand nous somme redirigé après la soumission de ce formulaire :



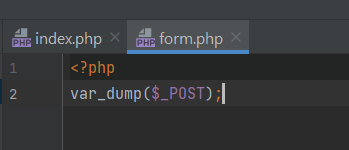
EN POST AUCUNE DONNEE VISIBLE

(formulaire avec la méthode GET)



EN GET JE VOIS TOUTES LES DONNEES

La page ou je reçois le formulaire EN POST (form.php)



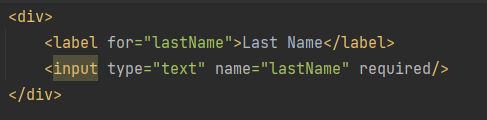
Résultat :



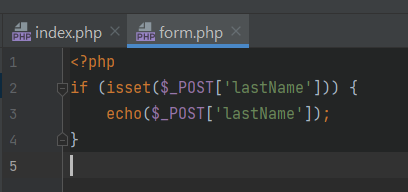
## La récupération des champs de formulaire en POST

Les champs en POST se récupèrent de la même manière qu’en GET en fonction de l’attribut name de nos champs HTML.

index.php

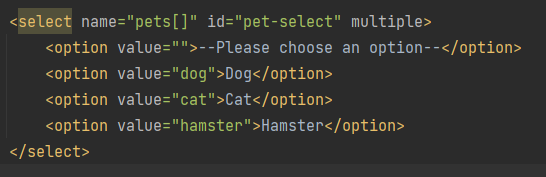


Post.php



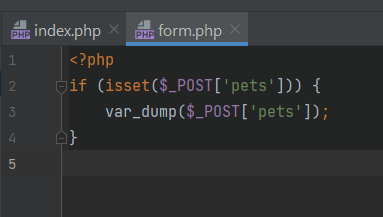
Vous pouvez récupérer tous les input HTML de la même façon, attention en cas de choix multiple, vous allez devoir les déclarer autrement :

Index.php



Le nom dans l’attribut name est suivi de « [] » afin d’indiquer à PHP que l’on va récupérer un tableau

form.php



Resultat :



## Exercices

### 1

* Créer un formulaire HTML sur une page, qui comprend comme champs :
  + firstName
  + lastName
  + age
  + sexe (champ radio, man ou woman)
  + job
  + hobbies (champ select multiple)
* Envoyer ce formulaire sur une autre page en mode PHP et récupérer tous les champs, puis les afficher

### 2

* En reprenant le formulaire créé précédemment, faites des vérifications sur les valeurs récupérées.
  + firstName : minimum : 2 caractères, maximum 20 caractères
  + lastName : minimum : 2 caractères, maximum 20 caractères
  + age : entre 0 et 99 ans
  + sexe : vérifier que la valeur envoyé est bien man ou woman
  + job : minimum 2 caractères, maximum 99 caractères
  + hobbies : vérifier que la personne est sélectionné au moins 2 passe-temps
* Si un des champs n’est pas bon après vérification, redirigé l’utilisateur sur une nouvelle page « error.php » et lui afficher un message d’erreur

### 3

* Maintenant que les vérifications sont mises en place, au lieu de rediriger l’utilisateur sur une autre page si il y a des erreurs, rediriger l’utilisateur sur la page ou est le formulaire et trouvez une solution pour afficher les erreurs en rouge en dessous des champs qui ont une erreurs.

### 4

* Créer une class « Personnage » qui va avoir les mêmes attributs que les champs de notre formulaire.
* Si notre formulaire est soumis et ne contient aucune erreur, construisez un objet à partir de notre class Personnage qui va recevoir comme valeur les résultats de notre formulaire et afficher à partir de notre objet les valeurs à notre utilisateur.

### 5

* Créer un nouveau projet et importer les class Hero crées lors du précédent cours.
* Créer un formulaire sur une page, qui aura un champ select pour le type de personnage à créer, un champ pour son nom et enfin un champ pour son lvl.
* Faites ensuite la fonctionnalité qui permet de créer des héros à partir de notre formulaire.