



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [PHP et HTML](#) ► [Validation des données de formulaire](#)

Validation des données de formulaire

À chaque fois que nous traiterons un formulaire, il faudra s'assurer de plusieurs choses :

1. que les attributs name des champs n'ont pas été modifiés
2. que les attributs value n'ont pas été modifiés
3. que les champs ont bien été remplis
4. que la valeur de ces champs correspond à ce qui est attendu et ne sont pas "nocives"

PHP propose une fonction de "filtrage" `filter_input()` qui nous permettra de tester si un champ existe, si ce champ a bien été rempli et si la valeur de ce champ correspond à ce que l'on attend. La fonction `filter_input()` prendra en premier paramètre une constante qui correspond au type d'input de notre formulaire : `INPUT_GET`, `INPUT_POST`, `INPUT_COOKIE`, `INPUT_SERVER` ou `INPUT_ENV`. Le deuxième paramètre sera le nom du champ, c'est-à-dire la valeur de son attribut name. Le troisième paramètre sera le [type de filtre](#) exprimée par une constante. Puis un tableau associatif d'options. Cette fonction renverra la valeur du champ si tout va bien, null si le champ n'existe pas, false si le champ est vide ou incorrect. Voici un exemple qui vérifiera si le champ de saisi nb est bien un entier compris entre 0 et 10 inclus.

```
1 <!DOCTYPE html>
2 <!--validation_formulaire_1.html-->
3 <html>
4   <head>
5     <meta charset="UTF-8">
6     <title></title>
7   </head>
8   <body>
9     <form action="validation_formulaire_2.php" method="post">
10      <input type="text" id="nb" name="nb"><br>
11      <input type="submit" value="Valider">
12    </form>
13  </body>
14 </html>
```

```
1 <?php
2
3 // validation_formulaire_2.php
4 $nb = filter_input(INPUT_POST, 'nb', FILTER_VALIDATE_INT,
5   ['options' => ['min_range' => 0, 'max_range' => 10]]);
6 if (!isset($nb)) {
7   echo 'le formulaire a été modifié';
8 } elseif ($nb === false) {
9   echo '$nb est vide ou incorrecte';
10 } else {
11   echo "\$nb est correcte et a pour valeur $nb";
12 }
```

La fonction `filter_input()` permet de valider des inputs selon plusieurs filtres : booléens, d'entiers, de décimaux, d'emails, d'adresses IP, d'URL ou d'expressions rationnelles.

Nous allons créer un formulaire dans une page `formulaire.php`. Nous y saisisons un login et un password. Ce formulaire aura pour cible le script `check_form.php` qui vérifiera si :

- le login contient des lettres, des chiffres ou des underscores et possède entre 4 et 12 caractères
- le password contient des lettres, des chiffres et possède entre 4 et 8 caractères

Si les champs sont valides, le script affichera : "le formulaire est OK !". Si les champs sont vides ou incorrects, le script ré-affichera le formulaire de saisie avec une liste des erreurs.

```

1 <!DOCTYPE html>
2 <!--validation_formulaire_3.php-->
3 <html>
4   <head>
5     <meta charset="UTF-8">
6     <title></title>
7   </head>
8   <body>
9     <?php if (isset($messages) && !empty($messages)) : ?>
10      <ul>
11        <?php foreach ($messages as $item) : ?>
12          <li><?php echo $item ?></li>
13        <?php endforeach; ?>
14      </ul>
15    <?php endif; ?>
16    <form action="validation_formulaire_4.php" method="post">
17      <label>
18        Login : <input type="text" id="login" name="login">
19      </label>
20      <label>Password : <input type="password" id="password"
21                        name="password">
22      </label>
23      <input type="submit" value="Valider">
24    </form>
25  </body>
26 </html>

```

À la première exécution, la variables messages n'existera pas et le contenu du if ne sera donc pas exécuté.

```

1 <?php
2
3 // validation_formulaire_4.php
4 $login = filter_input(INPUT_POST, 'login', FILTER_VALIDATE_REGEXP,
5   ['options' => ['regexp' => '/^\w{4,12}$/']]);
6 $password = filter_input(INPUT_POST, 'password', FILTER_VALIDATE_REGEXP,
7   ['options' => ['regexp' => '/^[a-zA-Z0-9]{4,8}$/']]);
8 $messages = [];
9 if (!$login) {
10   $messages[] = 'Le login est incorrect';
11 }
12 if (!$password) {
13   $messages[] = 'Le password est incorrect';
14 }
15 if (empty($messages)) {
16   echo 'le formulaire est OK !';
17 } else {
18   include './validation_formulaire_3.php';
19 }

```

Détaillons ce script :

```

1 <?php
2
3 $login = filter_input(INPUT_POST, 'login', FILTER_VALIDATE_REGEXP,
4   ['options' => ['regexp' => '/^\w{4,12}$/']]);

```

Cette expression assignera à login la valeur du champ si la valeur du champ est validée, false si la valeur du champ est invalide ou vide, null si le champ n'existe pas. La validation se fera selon une expression rationnelle, le type de filtre est donc FILTER_VALIDATE_REGEXP. Cette expression rationnelle est passée en argument dans un tableau associatif.

```

1 <?php
2
3 $password = filter_input(INPUT_POST, 'password', FILTER_VALIDATE_REGEXP,
4     ['options' => ['regexp' => '/^[a-zA-Z0-9]{4,8}$/']]);

```

Même principe de validation pour le password. La variable password vaudra donc la valeur du champ, ou false ou null.

```

1 <?php
2
3 $messages = [];

```

Nous créons un tableau qui contiendra les messages d'erreurs.

```

1 <?php
2
3 if (!$login) {
4     $messages[] = 'Le login est incorrect';
5 }

```

Si la valeur de login peut être évaluée à false alors nous ajoutons un message au tableau. À ce moment de l'exécution la valeur de la variable login peut être de trois types :

- si le champ est valide alors la valeur de login est la chaîne de caractère saisie par l'utilisateur
- si le champ est invalide, login a pour valeur false
- si le champ est vide, login a pour valeur null

Une chaîne de caractère non-vide est considérée comme vraie, les valeurs null et false sont considérées comme false. Donc si le champ est invalide ou manquant, la valeur de login pourra être considérée comme false et nous entrerons dans le if. Si la valeur de login est une chaîne de caractère non-vide, elle sera considérée comme vraie et nous n'entrerons pas dans le if. Dans le if, un message est ajouté au tableau messages.

```

1 <?php
2
3 if (!$password) {
4     $messages[] = 'Le password est incorrect';
5 }

```

Nous appliquons le même traitement à la variable password.

```

1 <?php
2
3 if (empty($messages)) {
4     echo 'le formulaire est OK !';
5 } else {
6     include './validation_formulaire_3.php';
7 }

```

Si le tableau de message est vide, cela signifie que les champs sont valides et que le formulaire est correct. Sinon nous incluons le script formulaire.php. Cette fois, dans formulaire.php, la variable messages existe bien puisqu'elle a été définie dans le script "incluant". Si cette variable n'est pas vide alors nous créons une liste non-ordonnée et nous bouclons sur le tableau pour afficher chaque item.

Les messages d'erreur peuvent aussi être affichés après les champs concernés, ou après, selon le design choisi. Par exemple :

```

1 <!DOCTYPE html>
2 <!-- validation_formulaire_5.php-->
3 <html>
4     <head>
5         <meta charset="UTF-8">
6         <title></title>

```

```

7      </head>
8      <body>
9          <form action="validation_formulaire_6.php" method="post">
10             <label>
11                 Login : <input type="text" id="login" name="login">
12             </label>
13             <?php if (isset($login_message)) echo $login_message ?><br>
14             <label>Password : <input type="password" id="password"
15                                 name="password">
16             </label>
17             <?php if (isset($password_message)) echo $password_message ?><br>
18             <input type="submit" value="Valider">
19         </form>
20     </body>
21 </html>

```

```

1 <?php
2
3 // validation_formulaire_6.php
4 $login = filter_input(INPUT_POST, 'login', FILTER_VALIDATE_REGEXP,
5     ['options' => ['regexp' => '/^\w{4,12}$/']]);
6 $password = filter_input(INPUT_POST, 'password', FILTER_VALIDATE_REGEXP,
7     ['options' => ['regexp' => '/^[a-zA-Z0-9]{4,8}$/']]);
8 $is_valid = true;
9 if (!$login) {
10     $login_message = 'Le login est incorrect';
11     $is_valid = false;
12 }
13 if (!$password) {
14     $password_message = 'Le password est incorrect';
15     $is_valid = false;
16 }
17 if ($is_valid) {
18     echo 'le formulaire est OK !';
19 } else {
20     include './validation_formulaire_5.php';
21 }

```

Fin

NAVIGATION



Accueil

■ Ma page

Pages du site

Mon profil

Cours actuel

Intro PHP

Participants

Le langage PHP : introduction

Les types et les variables

Les opérateurs

Les structures de contrôle

Les structure de données

Les fonctions

Les erreurs


Les fichiers










Les expressions rationnelles

PHP et HTML

 Introduction

 Syntaxe alternative

 T.P. tableau de personnes

-  [Les formulaires et les superglobales](#)
-  **[Validation des données de formulaire](#)**
-  [Détecter les modifications des formulaires](#)
-  [Affichage des données issues d'un formulaire](#)
-  [Les cookies](#)
-  [Les sessions](#)
-  [L'upload de fichier](#)
-  [T.P. formulaire](#)
-  [T.P. formulaire et session](#)

Petite application

[Mes cours](#)

ADMINISTRATION



[Administration du cours](#)

[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[Intro PHP](#)