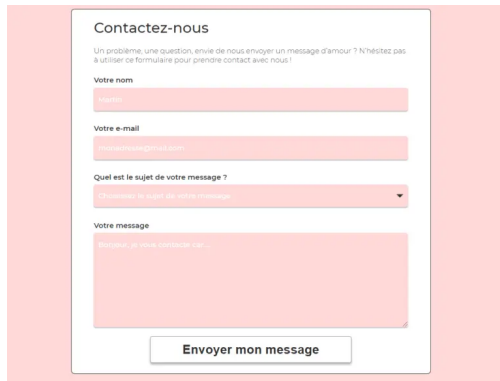


# La Validation des Formulaires avec PHP

La validation des formulaires en PHP est essentielle pour garantir la sécurité et l'intégrité des données soumises par les utilisateurs. Voici un guide qui explique comment valider les formulaires en PHP :

## 1. Conception du Formulaire HTML



Tout d'abord, concevons un formulaire HTML simple :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Formulaire de Validation</title>
</head>
<body>
  <form action="process_form.php" method="post">
    <label for="name">Nom :</label>
    <input type="text" id="name" name="name"
    required><br><br>
    <label for="email">Email :</label>
    <input type="email" id="email" name="email"
    required><br><br>

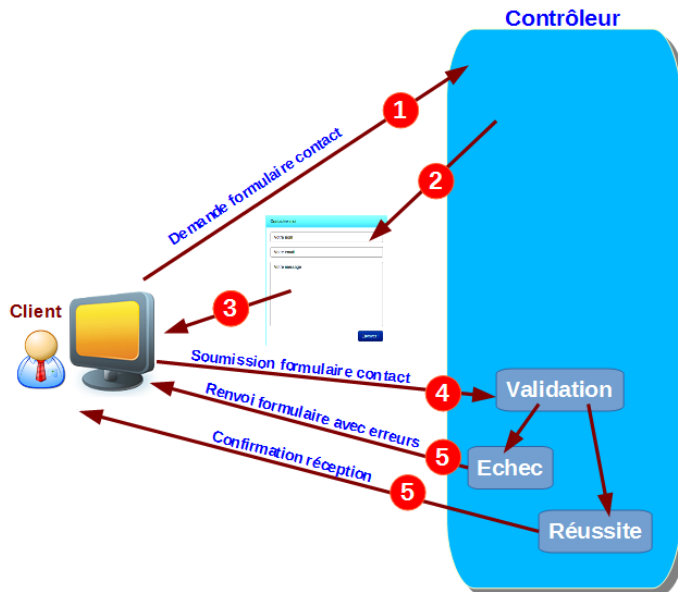
    <label for="age">Âge :</label>
    <input type="number" id="age" name="age"
    required><br><br>
```

```

        <input type="submit" value="Envoyer">
    </form>
</body>
</html>

```

## 2. Traitement et Validation des Données en PHP



Le fichier `process_form.php` va traiter et valider les données soumises par le formulaire :

```

<?php
// Fonction de nettoyage des données
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

// Initialisation des variables d'erreur et
des variables de données

```

```

$nameErr = $emailErr = $ageErr = "";
$name = $email = $age = "";

// Vérification si le formulaire a été soumis
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Validation du nom
    if (empty($_POST["name"])) {
        $nameErr = "Le nom est requis";
    } else {
        $name = test_input($_POST["name"]);
        // Vérification si le nom contient uniquement
        // des lettres et des espaces
        if (!preg_match("/^[a-zA-Z-'\ ]*$/", $name)) {
            $nameErr = "Seules les lettres et les
            espaces sont autorisés";
        }
    }
}

// Validation de l'email
if (empty($_POST["email"])) {
    $emailErr = "L'email est requis";
} else {
    $email = test_input($_POST["email"]);
    // Vérification si l'adresse email est bien
    // formée
    if (!filter_var($email,
        FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Format d'email invalide";
    }
}

// Validation de l'âge
if (empty($_POST["age"])) {
    $ageErr = "L'âge est requis";
} else {
    $age = test_input($_POST["age"]);
    // Vérification si l'âge est un nombre positif
    if (!filter_var($age, FILTER_VALIDATE_INT,

```

```

        ["options" => ["min_range" => 1]])) {
    $ageErr = "L'âge doit être un nombre
    positif";
}
}

// Si aucune erreur, traitement des données
if (empty($nameErr) && empty($emailErr)
    && empty($ageErr)) {
    // Code pour traiter les données, par exemple,
    les stocker dans une base de données
    echo "Nom : $name<br>";
    echo "Email : $email<br>";
    echo "Âge : $age<br>";
} else {
    // Affichage des erreurs
    echo "Erreurs dans le formulaire :<br>";
    echo $nameErr . "<br>";
    echo $emailErr . "<br>";
    echo $ageErr . "<br>";
}
}
?>

```

## Explication du Code

1. **test\_input()** : Cette fonction nettoie les données en supprimant les espaces en début et fin de chaîne, les barres obliques inverses, et en convertissant les caractères spéciaux en entités HTML pour éviter les injections de code.
2. **Validation des données** :
  - Pour le champ **nom**, on vérifie s'il n'est pas vide et s'il ne contient que des lettres et des espaces.
  - Pour le champ **email**, on utilise le filtre `FILTER_VALIDATE_EMAIL` pour valider le format de l'email.
  - Pour le champ **âge**, on s'assure qu'il s'agit d'un entier positif.

3. **Affichage des Erreurs** : Si des erreurs sont détectées, elles sont affichées à l'utilisateur. Sinon, les données sont traitées (par exemple, en les affichant ou en les stockant dans une base de données).

### 3. Sécurité Additionnelle

Pour renforcer la sécurité : - Utilisez des **jetons CSRF** pour éviter les attaques par falsification de requête intersite. - Évitez les **injections SQL** en utilisant des instructions préparées avec des paramètres liés lors de l'interaction avec une base de données.

### Conclusion

La validation des formulaires est cruciale pour sécuriser votre application web et garantir la qualité des données. Ce guide couvre les bases de la validation en PHP, mais n'hésitez pas à adapter et à étendre ces concepts selon vos besoins spécifiques.