Développement PHP
Partie 3 : Transmettre des données

DENIS LOEUILLET – IFA - 2017

Partie 3 : Transmettre des données

- Variables superglobales
- Transmettre des données avec l'URL
- Transmettre des données avec les formulaires
- Sessions & Cookies
- Lire et écrire dans un fichier

Partie 3 : Transmettre des données
Avec les formulaires

Les formulaires constituent le principal moyen pour vos visiteurs d'entrer des informations sur votre site.

Les formulaires permettent de créer une interactivité.

On a besoin des formulaires partout pour échanger des informations avec nos visiteurs.

PHP et le HTML sont intimement liés.

- HTML permet de créer le formulaire,
- PHP permet de traiter les informations que le visiteur a entrées dans le formulaire.

Partie 3 : Transmettre des données Avec les formulaires

1. Créer la base du formulaire
2. Les éléments du formulaire
3. L'envoi de fichiers
4. Ne faites jamais confiance aux données reçues

Partie 3 : Transmettre des données Avec les formulaires

1. Créer la base du formulaire
   - En HTML, pour insérer un formulaire, on se sert de la balise <form>

```
1 <form method="post" action="cible.php">
2
3 <p>
4     On insèrera ici les éléments de notre formulaire.
5 </p>
6
7 </form>
```

   - On écrira le contenu de notre formulaire entre les balises <form> et </form>.
   - Il y a deux attributs très importants à connaître pour la balise <form> :
     - ✓ la méthode (method)
     - ✓ la cible (action).

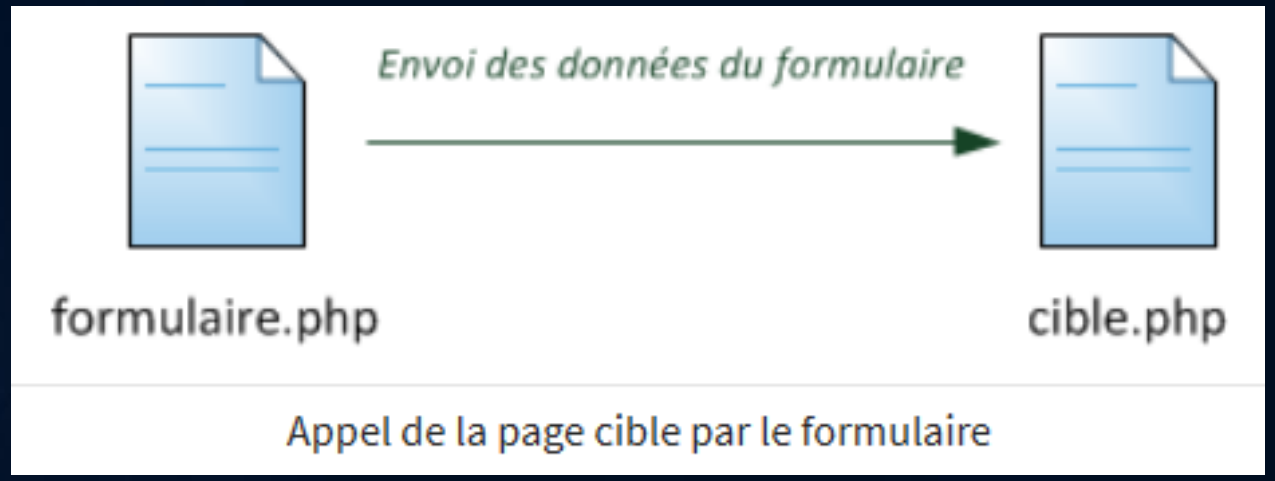
Partie 3 : Transmettre des données
Avec les formulaires : créer la base du formulaire

- La méthode
  Il existe plusieurs moyens d'envoyer les données du formulaire (plusieurs méthodes)

  - get :
    - ✓ les données transiteront par l'URL.
    - ✓ On pourra les récupérer grâce à l'array $_GET.
    - ✓ Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL (il est préférable de ne pas dépasser 256 caractères).
  - post :
    - ✓ les données ne transiteront pas par l'URL,
    - ✓ l'utilisateur ne les verra donc pas passer dans la barre d'adresse.
    - ✓ Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent.
    - ✓ Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier si tous les paramètres sont bien présents et valides.
    - ✓ On ne doit pas plus faire confiance aux formulaires qu'aux URL.

Partie 3 : Transmettre des données
Avec les formulaires : créer la base du formulaire

- La cible
  L'attribut « action » sert à définir la page appelée par le formulaire.
  C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.



Appel de la page cible par le formulaire

- Le formulaire se trouve dans la page formulaire .php
- une fois le formulaire envoyé (lorsqu'on a cliqué sur le bouton « Valider »), le visiteur est redirigé vers la page cible.php qui reçoit les données du formulaire

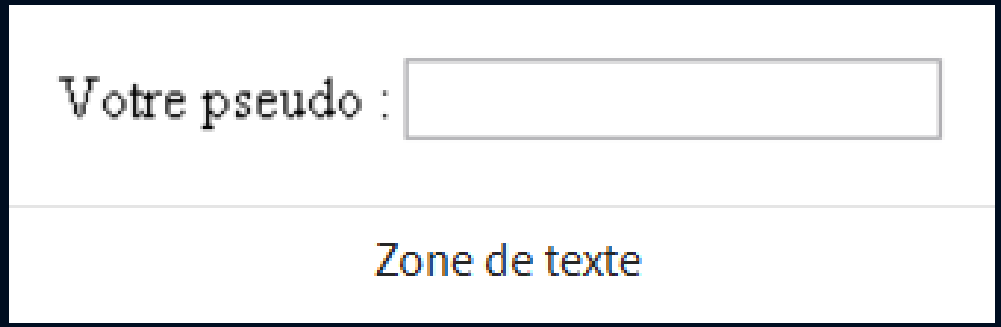
Partie 3 : Transmettre des données Avec les formulaires

2. Les éléments du formulaire
   - Dans un formulaire, on peut insérer beaucoup d'éléments différents :
     - zones de texte,
     - boutons,
     - cases à cocher,
     - listes déroulantes,
     - les champs cachés,

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les petites zones de texte
  - Une zone de texte ressemble à la figure suivante
  - En HTML on l'insère avec la balise « input » :

```
1  <input type="text" />
```



- ✓ Il y a deux attributs à connaître :
  - ❖ name (obligatoire) : c'est le nom de la zone de texte, c'est lui qui va produire une variable.
    <input type="text" name="pseudo" />.
  - ❖ value (facultatif) : c'est ce que contient la zone de texte au départ. Par défaut, la zone de texte est vide mais il peut être pratique de pré-remplir le champ.
    <input type="text" name="pseudo" value="M@teo21" />
- ✓ Le texte que le visiteur aura saisi sera disponible dans la page cible sous la forme d'une variable $_POST['pseudo']

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les petites zones de texte
  - Créer un formulaire qui demande le prénom du visiteur puis qui l'affiche sur la page cible.php.
    - ✓ On va donc distinguer deux codes source :
      - celui de la page du formulaire
      - celui de la page cible.
    - ✓ Voici le code de la page formulaire.php :
      - le champ <input type="submit" /> permet de créer le bouton de validation du formulaire qui commande l'envoi des données, et donc la redirection du visiteur vers la page cible.

```
1  <p>
2      Cette page ne contient que du HTML.<br />
3      Veuillez taper votre prénom :
4  </p>
5
6  <form action="cible.php" method="post">
7  <p>
8      <input type="text" name="prenom" />
9      <input type="submit" value="Valider" />
10 </p>
11 </form>
```

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les petites zones de texte
  - Créer un formulaire qui demande le prénom du visiteur puis qui l'affiche sur lapage cible.php.
    - ✓ Voici le code de la page cible.php :
      - ❖ Cette page va recevoir le prénom dans une variable nommée $_POST['prenom']

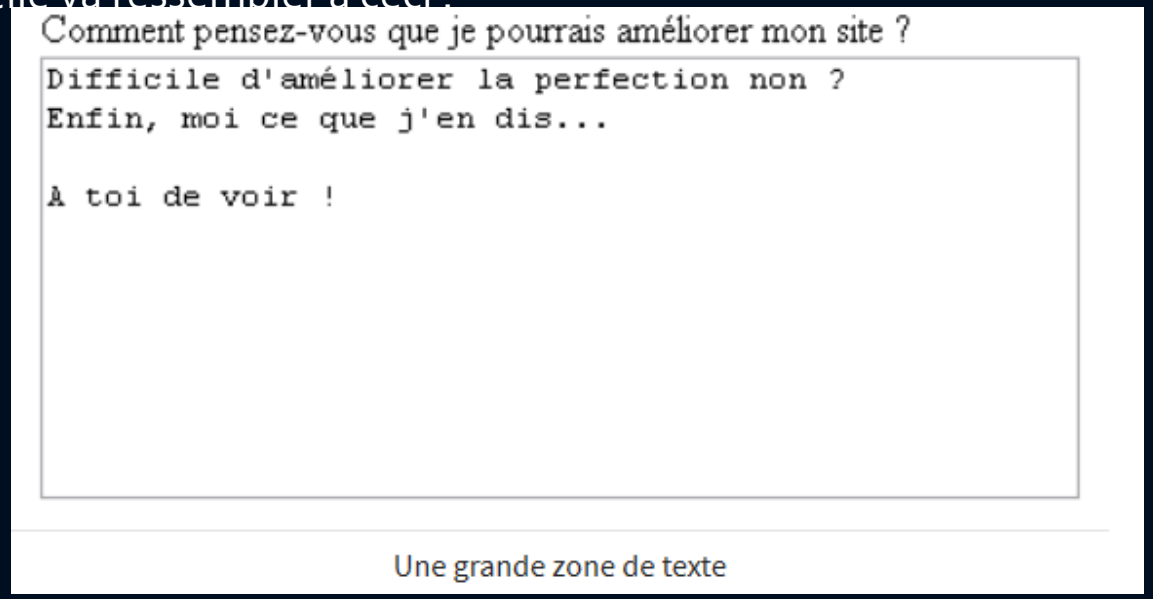
```
1 <p>Bonjour !</p>
2
3 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo $_POST['prenom']; ?> !</p>
4
5 <p>Si tu veux changer de prénom, <a href="formulaire.php">clique ici</a> pour revenir à la
  page formulaire.php.</p>
```

      - ❖ Dans cible.php on a affiché une variable $_POST['prenom'] qui contient ce que l'utilisateur a entré dans le formulaire.

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les grandes zones de texte
  - La grande zone de texte est nommée « textarea », elle va ressembler à ceci :
  - On peut y écrire autant de lignes que l'on veut
  - Adapté pour les textes longs
  - Le code HTML associé :

```
1 <textarea name="message" rows="8" cols="45">
2 Votre message ici.
3 </textarea>
```



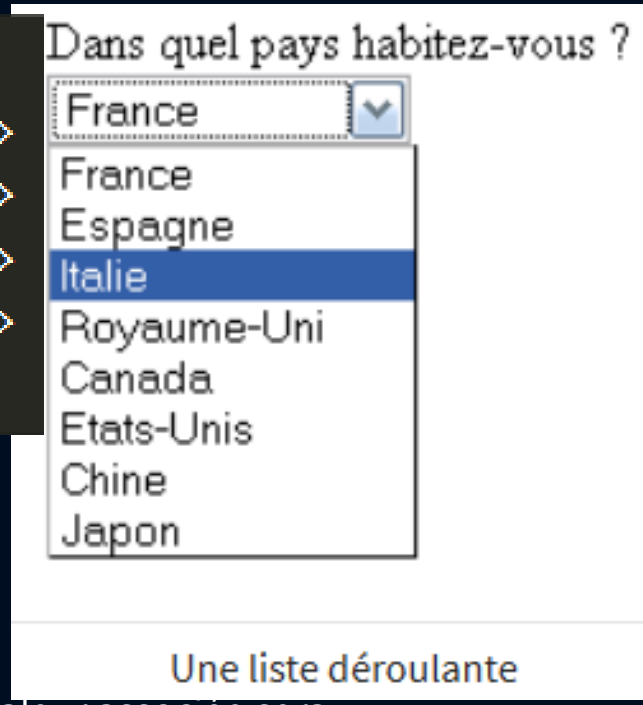
Une grande zone de texte

  - Là encore plusieurs attributs associés :
    - ✓ name : qui va définir le nom de la variable qui sera créée dans le fichier cible
    - ✓ Pas d'attribut value : En fait, le texte par défaut est ici écrit entre le<textarea>et le</textarea>. Si vous ne voulez rien mettre par défaut, alors n'écrivez rien entre<textarea>et</textarea>. C'est ce texte qui sera utilisé comme valeur
    - ✓ Les attributs « rows » et « cols » permettent de définir la taille de la zone de texte en hauteur et en largeur respectivement.

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- La liste déroulante
  - Code HTML associé :
  - Balise « select »
  - Attribut « name »
  - Chaque balise « option » à l'intérieur de la balise « select » représente un des items de la liste déroulante
  - La variable créée dans le fichier cible à partir de l'attribut « name » contiendra le choix qu'aura fait l'utilisateur :
    - ✓ Si l'utilisateur sélectionne « Choix 3 », la variable créée sera $_POST['choix'] et la valeur associée sera « choix3 »
  - Pour sélectionner un choix par défaut il suffit d'ajouter l'attribut « selected » à l'option choisie par défaut, ce qui va donner :

```
1 <select name="choix">
2     <option value="choix1">Choix 1</option>
3     <option value="choix2">Choix 2</option>
4     <option value="choix3">Choix 3</option>
5     <option value="choix4">Choix 4</option>
6 </select>
```



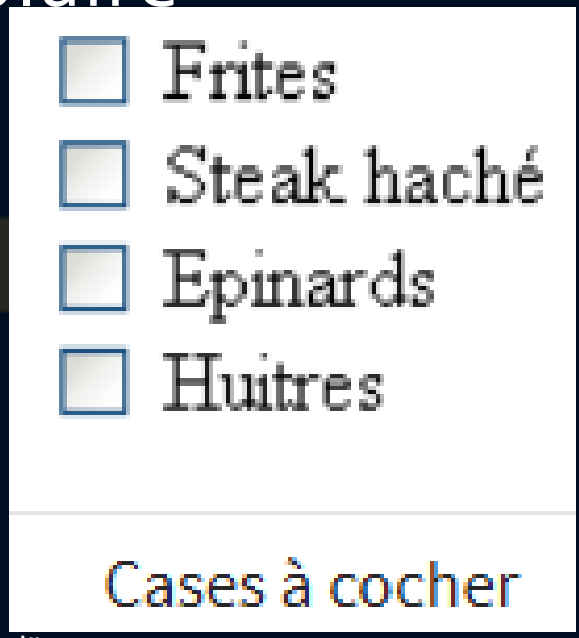
Une liste déroulante

```
1 <option value="choix3" selected="selected">Choix 3</option>
```

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les case à cocher
  - Code HTML associé :

```
1 <input type="checkbox" name="case" id="case" /> <label for="case">Ma case à cocher</label>
```



- ✓ Attribut « name » qui va générer une variable dans le fichier cible $_POST[‘case’]
- ✓ Valeurs associées :
  - ❖ Si la case est cochée : $_POST['case'] vaudra « on »
  - ❖ Si la case n’est pas cochée alors $_POST['case'] n'existera pas. Vous pouvez faire un test avec isset($_POST['case']) pour vérifier si la case a été cochée ou non.
  - ❖ Si vous voulez que la case soit cochée par défaut, il faudra lui rajouter l'attribut checked="checked".

```
1 <input type="checkbox" name="case" checked="checked" />
```

- ✓ L'utilisation de la balise <label> n'est pas obligatoire mais je la recommande fortement. Elle permet d'associer le libellé à la case à cocher qui a le même « id » que son attribut « for », ce qui permet de cliquer sur le libellé pour cocher la case. On y gagne donc en ergonomie.

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les boutons d'option
  - Code HTML associé :


```
1 Aimez-vous les frites ?
2 <input type="radio" name="frites" value="oui" id="oui" checked="checked" /> <label
  for="oui">Oui</label>
3 <input type="radio" name="frites" value="non" id="non" /> <label for="non">Non</label>
```

- ✓ Ici, les 2 boutons d'option ont le même nom (frites)
  - ❖ Très important car les boutons fonctionnent par groupes :
    - ○ tous les boutons d'option d'un même groupe doivent avoir le même nom.
    - ○ Cela permet au navigateur de savoir lesquels désactiver quand on active un autre bouton du groupe.
    - ○ Il serait bête en effet de pouvoir sélectionner à la fois « Oui » et « Non ».
- ✓ Pour pré-cocher l'un de ces boutons, faites comme pour les cases à cocher : rajoutez un attribut checked="checked". Ici, comme vous pouvez le voir, « Oui » est sélectionné par défaut.
- ✓ Dans le fichier cible, une variable$_POST['frites'] sera créée. Elle aura la valeur du bouton d'option choisi par le visiteur, issue de l'attribut « value ». Si on aime les frites, alors on aura $_POST['frites'] = 'oui'.
- ✓ Ne pas oublier de renseigner l'attribut « value » du bouton d'option car c'est lui qui va déterminer la valeur de la variable.

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Les champs cachés
  - Code HTML associé :

```
1  <input type="hidden" name="pseudo" value="Mateo21" />
```

  - Les champs cachés constituent un type de champ à part.
    - ✓ C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur,
    - ✓ Il qui va quand même créer une variable avec une valeur.
    - ✓ On peut s'en servir pour transmettre des informations fixes.
    - ✓ Exemple :
      - Si vous voulez retenir que le pseudo du visiteur est « Mateo21 »
      - À l'écran, sur la page web on ne verra rien.
      - Mais dans le fichier cible, une variable $_POST['pseudo'] sera créée, et elle aura la valeur « Mateo21 » !
      - C'est apparemment inutile, mais vous verrez que vous en aurez parfois besoin.
    - ✓ Ce champs est caché mais il peut tout de même être visible si on affiche le code source de la page et on peut même avec des outils appropriés en modifier la valeur!!!
      - Ne faites jamais confiance aux données transmises par les visiteurs

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Exercice :

  Créez un formulaire dans un fichier formulaire.html :

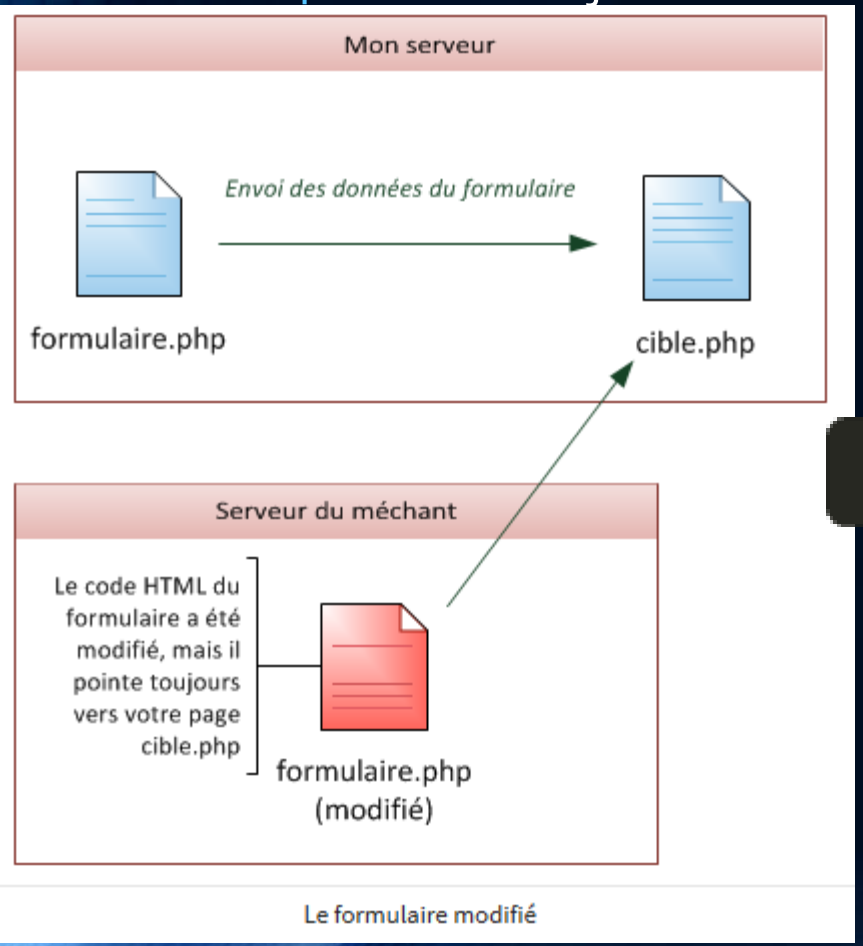
  - Ce formulaire doit contenir :
    - 3 zones de texte pour saisir
      - ✓ Votre nom
      - ✓ Votre prénom
      - ✓ Votre âge
    - Une liste de choix pour sélectionner le sexe
  - Le fichier cible sera nommé cible.php et devra effectuer les actions suivantes:
    - ✓ Récupère les variables du formulaire
    - ✓ Teste l'existence des variables
    - ✓ Teste le type des variables reçues
    - ✓ Teste les valeurs des différentes variables
    - ✓ Affiche un message contenant les valeurs des données reçues si celles-ci sont correctes
    - ✓ Affiche un message différent si les données reçues ne sont pas correctes

Partie 3 : Transmettre des données
Avec les formulaires

4. Ne faites jamais confiance aux données reçues
   - Les données reçues par le POST ne sont pas plus fiables que les données reçues par le GET
   - Pourquoi les formulaires ne sont pas sûrs ?
     - ✓ Tout ce que nous avons dit sur les URL reste valable ici
     - ✓ Toutes les informations qui proviennent de l'utilisateur, à savoir les données de $_GET et de $_POST, doivent être traitées avec la plus grande méfiance.
     - ✓ Vous ne pouvez pas supposer que vous allez recevoir ce que vous attendiez.
       - L'utilisateur peut se tromper de bonne foi
       - L'utilisateur peut essayer de vous tromper
       - Même si les données sont issues d'un formulaire
     - ✓ Les visiteurs ne peuvent pas modifier vos pages web sur le serveur … par contre, ils peuvent les reprendre et les modifier ailleurs!!
       - Le code PHP n'est jamais visible par vos visiteurs
       - Le code HTML (du formulaire par exemple) peut être vu par tout le monde
       - Quelqu'un peut créer une copie modifiée de votre formulaire et la stocker sur son propre serveur

Partie 3 : Transmettre des données
Avec les formulaires

4. Ne faites jamais confiance aux données reçues



Le formulaire modifié

Sur le schéma, le « méchant » a pris le code HTML de votre formulaire, l'a modifié et l'a enregistré sur son serveur (ou même sur son ordinateur).

L'attribut « action » a été modifié pour indiquer l'adresse absolue (donc complète) de votre page cible :

```
1 <form method="post" action="http://www.monsite.com/cible.php">
```

Le méchant peut maintenant modifier votre formulaire, ajouter des champs, en supprimer, bref faire ce qu'il veut avec !

Votre page cible.php n'y verra que du feu car il est impossible de savoir avec certitude de quel formulaire vient le visiteur.

les formulaires sont modifiables par tous les visiteurs

Partie 3 : Transmettre des données
Avec les formulaires

4. Ne faites jamais confiance aux données reçues
   - Il y a un moyen encore plus simple de modifier le formulaire de votre site sans avoir accès à votre serveur.
   - Les navigateurs embarquent des « outils pour les développeurs » qui permettent de modifier le code HTML de la page que l'on visite en temps réel.
   - Firefox peut faire ça facilement avec son célèbre plugin Firebug.

Partie 3 : Transmettre des données Avec les formulaires

4. Ne faites jamais confiance aux données reçues : la faille XSS
   - La faille XSS (pour cross-site scripting) est très ancienne et on la trouve encore sur de nombreux sites web, même professionnels !
   - C'est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages pour le faire exécuter à vos visiteurs.
   - Exemple :

```
1 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo $_POST['prenom']; ?>
  !</p>
```

     - ✓ Si le visiteur décide d'écrire du code HTML à la place de son prénom, cela fonctionnera très bien !
     - ✓ Par exemple, imaginons qu'il écrive dans le champ « Prénom » le code :Badaboum. Le code source HTML qui sera généré par PHP sera le suivant :

```
1 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <strong>Badaboum</strong> !</p>
```

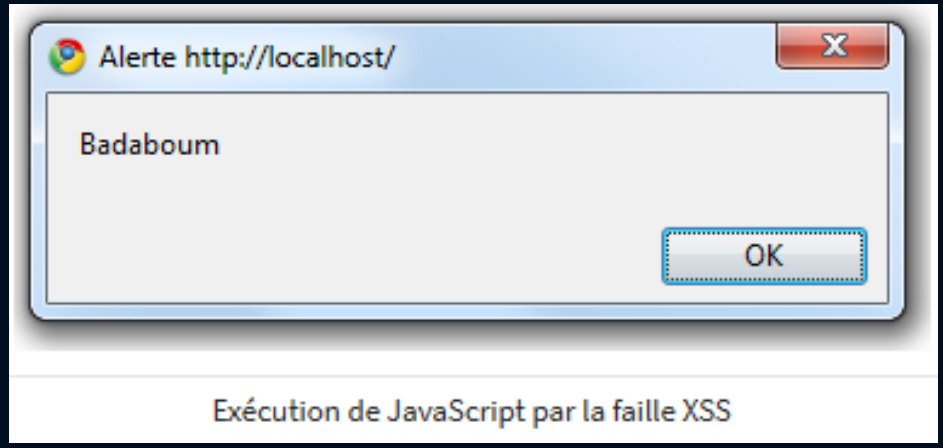
     - ✓ Il peut aussi ouvrir des balises de type <script> pour faire exécuter du code JavaScript au visiteur qui visualisera la page !

```
1 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <script
  type="text/javascript">alert('Badaboum')</script> !</p>
```

Partie 3 : Transmettre des données Avec les formulaires

4. Ne faites jamais confiance aux données reçues : la faille XSS

- Tous les visiteurs qui arriveront sur cette page verront une boîte de dialogue JavaScript s'afficher. Plutôt gênant.
- Les choses deviennent vraiment critiques si le visiteur veut vraiment vous nuire!!!!
- Résoudre le problème est facile :
  - ✓ Il faut protéger le code HTML en l'échappant, c'est-à-dire en affichant les balises (ou en les retirant) plutôt que de les faire exécuter par le navigateur.
  - ✓ Pour échapper le code HTML, il suffit d'utiliser la fonction « htmlspecialchars » (http://php.net/manual/fr/function.htmlspecialchars.php) qui va transformer les chevrons des balises HTML<>en < et > respectivement. Cela provoquera l'affichage de la balise plutôt que son exécution.



Exécution de JavaScript par la faille XSS

```
1 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo
  htmlspecialchars($_POST['prenom']); ?> !</p>
```

```
1 Je sais comment tu t'appelles, hé hé. Tu t'appelles <strong>Badaboum</strong> !
```

Partie 3 : Transmettre des données
Avec les formulaires

4. Ne faites jamais confiance aux données reçues : la faille XSS
   - Il faut penser à utiliser cette fonction sur tous les textes envoyés par l'utilisateur qui sont susceptibles d'être affichés sur une page web.
   - Tout ce qui est affiché et qui vient à la base d'un visiteur, vous devez penser à le protéger avec « htmlspecialchars »
   - Si vous préférez retirer les balises HTML que le visiteur a tenté d'envoyer plutôt que de les afficher, utilisez la fonction « strip_tags » (http://php.net/manual/fr/function.strip-tags.php)

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Exercice :

  Modifiez votre formulaire.html :

  - Ce formulaire doit contenir :
    - 3 zones de texte pour saisir
      - ✓ Votre nom
      - ✓ Votre prénom
      - ✓ Votre âge
    - Une liste de choix pour sélectionner le sexe
  - Le fichier cible sera nommé cible.php et devra effectuer les actions suivantes:
    - ✓ Récupère les variables du formulaire
    - ✓ Protéger votre code contre les injection de code HTML ou JAVASCRIPT
    - ✓ Teste l'existence des variables
    - ✓ Teste le type des variables reçues
    - ✓ Teste les valeurs des différentes variables
    - ✓ Affiche un message contenant les valeurs des données reçues si celles-ci sont correctes
    - ✓ Affiche un message différent si les données reçues ne sont pas correctes

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Exercice :
  - Essayez de modifier votre formulaire depuis votre navigateur
    - ✓ Modifiez le noms des divers champs du formulaire
    - ✓ Modifiez la cible du formulaire
    - ✓ Modifiez les valeurs des champs

Partie 3 : Transmettre des données
Avec les formulaires : les éléments du formulaire

- Exercice :
  - Réalisez une fonction qui va servir à contrôler les données issues des données reçues par $_GET et $_POST, cette fonction aura pour rôle de :
    - ✓ Convertir les variables dans le type attendu
    - ✓ Protéger les données contre les injections de code
    - ✓ Être utilisable pour tous types de données :
      - ❖ STRING
      - ❖ INT
      - ❖ FLOAT
      - ❖ BOOL
    - Syntaxe à l'utilisation :
      - ✓ $variable_controlee = ma_function_de_controle ('type_superglobale', 'nom_variable', 'type_variable');
        - ❖ Exemple : $variable_controlee = ma_fonction_de_controle ('POST', 'nom', 'STRING');