VIII) Formulaires et transmission de données

PHP - Initiation

Formulaires et transmission de données

VIII.1) Formulaires

Les formulaires sont un outil primordial lors du développement d'un applicatif Web. C'est grace à eux que l'utilisateur va pouvoir transmettre des informations et ainsi réaliser n'importe quelle action pouvant être exécutée par un script.

VIII.1.1) Balise FORM

Tout formulaire HTML commence et se termine par cette balise. Les champs permettant de transmettre les informations seront à l'intérieur.

Syntaxe

```
<form>
   champs
</form>
```

PHP - Initiation

Formulaires et transmission de données

Ces champs (aussi appelés 'widgets') peuvent prendre différentes formes : zones de texte, menus, cases à cocher...
Afin de transmettre son information, chaque widget possède un **nom** auquel est associé une **valeur**. (Un peu comme une variable)

VIII.1.2) Balise INPUT TEXT

<input type='text' name='*montexte*'>

Mis à part les attributs **type** et **name** (obligatoires si l'on veut transmettre correctement) il en existe de très nombreux autres.

value	initialise la valeur du champ
size	modifie la largeur du champ (nbr de caractères)
maxlength	limite le nombre de caractères pouvant être renseignés
readonly	empêche la modification du contenu ('true' ou 'false')

VIII.1.3) Balise INPUT PASSWORD

<input type='password' name='*motdepasse*'>

C'est une variante à la balise text qui permet de passer un mot de passe : le comportement et les caractéristiques sont les mêmes mais le contenu tapé par l'utilisateur n'apparait pas en clair, chaque caractères étant remplacé par un petit symbole.

VIII.1.4) Balise TEXTAREA

<textarea name='*longtexte*'><textarea>

C'est une zone de texte multiligne. Les données à transmettre sont contenues entre les 2 balises.

rows	permet de modifier la taille en nombre de lignes
cols	permet de modifier la taille en nombre de colonnes

PHP - Initiation

Formulaires et transmission de données

VIII.1.5) Balise SELECT

```
<select name='maliste'>
   <option value='valeur1'></option>
   <option value='valeur2'></option>
   <option value='valeur3'></option>
   ...
</select>
```

Ce widget permet de construire des menus déroulants. Le nom est défini dans la balise **SELECT** tandis que les différentes valeurs possibles sont passées dans les balises **OPTION**. Les textes illustrant les choix possibles sont eux placés à l'intérieur des balises.

selected permet de prédéfinir un choix ('true')

PHP - Initiation

Formulaires et transmission de données

Par défaut un seul choix peut être fait par l'utilisateur. Il est cependant possible de permettre des choix multiples avec l'attribut **multiple** que l'on déclare dans la balise **SELECT**.

multiple	permet d'autoriser les choix multiples ('true')
size	détermine combien d'options sont visibles par défaut

Dans le cas de choix multiples, les valeurs des choix seront envoyées sous forme de tableau. Nous devons donc rajouter des **[]** (crochets) au nom de notre **SELECT**.

```
<select name='maliste[]' multiple='true'>
   <option value='valeur1'></option>
   <option value='valeur2'></option>
   <option value='valeur3'></option>
   ...
</select>
```

VIII.1.6) Balise INPUT CHECKBOX

<input type='checkbox' name='*macheckbox*'>

Permet de réaliser une case à cocher. Chaque clic permet de cocher/décocher le widget. **Attention** : seules les valeurs des cases cochées sont transmises ! Si la valeur n'est pas précisée, la valeur transmise sera 'on'. Il est possible d'envoyer plusieurs valeurs associées au nom de notre widget. Pour cela on utilise le même nom dans nos cases en ajoutant des **[]** (crochets).

<input type='checkbox' name='*macheckbox*[]' value='*val1*'>
<input type='checkbox' name='*macheckbox*[]' value='*val2*'>
<input type='checkbox' name='*macheckbox*[]' value='*val3*'>

checked permet de précocher des cases ('true')

PHP - Initiation

Formulaires et transmission de données

VIII.1.7) Balise INPUT RADIO

<input type='radio' name='*monradio*'>

Ce widget seul n'a aucun interêt, il s'utilise tout le temps en configuration 'multiple'. En effet seul, le bouton radio une fois coché ne peut plus se décocher. En l'associant à d'autre, il permet de faire un choix unique entre eux en 'switchant'.

Le choix étant unique, pas besoin ici des **[]** (crochets).

<input type='radio' name='*monradio*' value='*val1*'>
<input type='radio' name='*monradio*' value='*val2*'>
<input type='radio' name='*monradio*' value='*val3*'>

checked permet de précocher un radio ('true')

PHP - Initiation

Formulaires et transmission de données

VIII.1.8) Balise INPUT HIDDEN

<input type='hidden' name='*mavalinvisible*' value='*val*'>

Ce widget permet de transmettre une information de notre choix de manière 'invisible'.

Attention : le widget n'apparait pas pour l'utilisateur mais s'il observe le code de la page le contenant, il pourra tout à fait lire la donnée transmise. Ce widget ne doit donc pas être utilisé pour faire passer des informations cruciales nécessitant un minimum de sécurité.

On l'utilisera pour transmettre toute donnée ne nécessitant pas un choix de la part de l'utilisateur.

PHP - Initiation

Formulaires et transmission de données

VIII.1.9) Balises INPUT SUBMIT, RESET et BUTTON

<input type='submit' value='*envoyer*'>

Ce widget fait apparaitre un bouton dont le clic envoi le formulaire.

<input type='reset' value='*effacer*'>

Ce widget fait apparaitre un bouton donc le clic réinitialise le formulaire à ses valeurs par défaut.

<input type='button' value='*bouton*'>

Ce widget fait apparaitre un bouton neutre : un clic sur ce dernier ne génère aucune action. Son interêt est que nous pourrons lui associer du code (JS ou jQuery en général) qui nous permettra de lancer une action de notre choix.

PHP - Initiation

Formulaires et transmission de données

VIII.2) Transmission de données

Maintenant que nous savons construire un formulaire et récupérer des informations auprès de l'utilisateur, voyons comment transmettre ces dernières à un script PHP qui pourra les traiter.

VIII.2.1) Destination

C'est à l'intérieur de la balise **FORM** que nous allons préciser où envoyer ces données grace à l'attribut **action**.

<form action='*fichier*'>

Notes :
- Si l'on ne précise pas le fichier de destination, les données sont envoyées à la page elle-même.
- Il est tout à fait possible d'appeller un script sur un autre serveur.

PHP - Initiation

Formulaires et transmission de données

VIII.2.2) Méthode GET

Il existe principalement 2 façons de transmettre les données. Celle par défaut s'appelle **GET**. (Elle s'applique si l'on ne précise rien)

<form action='*fichier*' method='get'>

Les données sont transmises 'en clair' dans l'URL, c'est à dire qu'on peut les lire sans problèmes à travers ce qu'on appelle la '**query string**'. Elle commence par un **?** (point d'interrogation) juste après le nom du fichier de destination puis suivent les couples ***nom=valeur*** séparés par des **&** (esperluette).

Exemple :

mon_script.php?banane=4&pomme=1&ananas=3

PHP - Initiation

Formulaires et transmission de données

Cette méthode est souvent utilisée pour débuguer ou tester son code mais rarement pour transmettre des données et ce, principalement pour 2 raisons :

- **Manque de sécurité** : toutes les informations transmises sont lisibles dans l'URL.
- **Limitation de taille** : les navigateurs limitent la taille maximum de la 'query string'. (en général entre 256 et 512 caractères)
- **URL encodage** : certains caractères doivent être encodés dans l'URL. Cela est fait automatiquement par le formulaire mais si l'on souhaite nous même la générer, les choses se compliquent.

Exemples :

prenom=Éric	devient	*prenom=%C3%89ric*
texte=Bonjour à tous	devient	*texte=Bonjour+%C3%A0+tous*
operateur=b&you	devient	*operateur=b%26you*

PHP - Initiation

Formulaires et transmission de données

VIII.2.3) Méthode POST

<form action='*fichier*' method='post'>

Avec cette méthode, les données sont transmises dans l'entête HTTP de manière masquée. **Attention** : masquée ne veut pas dire cryptée (comme avec le protocole HTTPS) ce qui signifit qu'une personne mal intentionnée et qui s'y connait un minimum pourra intercepter et lire vos informations.

De plus cette méthode permet d'envoyer en théorie une quantité d'informations illimitée et c'est la seule qui autorise l'envoi de fichiers en précisant dans la balise **form** un nouvel attribut pour l'encodage : **enctype**

<form action='*fichier*' method='post' enctype='multipart/form-data'>

PHP - Initiation

Formulaires et transmission de données

VIII.2.4) Récupération

La récupération des données en provenance d'un formulaire est d'une simplicité déconcertante. Il existe des genres de super variables qui contiennent tout simplement ces données.

$_GET

Fonctionne comme un tableau associatif dont chaque mot-clé correspond à un nom de champ (**name**) et auquel est bien évidemment associé la valeur correspondante.

Il existe l'équivalent pour les données passées en POST : **$_POST**

Il existe même une variante qui contient les 2 : **$_REQUEST**
(Cette dernière contient également les infos des cookies mais laissons ça de côté pour l'instant...)

PHP - Initiation

Formulaires et transmission de données

Exemple :

HTML

```
<form action='traitement.php' method='get'>
	<input type='text' name='prenom'>
	<input type='text' name='nom'>
	<input type='submit' value='ENVOI'>
</form>
```

PHP

```
<?php
   $prenom = $_GET['prenom'];
   $nom = $_GET['nom'];
   echo 'Bonjour '.$prenom.' '.$nom;
?>
```

Cela fonctionnera exactement de la même manière avec **$_POST** ou **$_REQUEST**.

PHP - Initiation

Formulaires et transmission de données

Certaines données peuvent être envoyées sous forme de tableau. C'est le cas par exemple pour des checkbox associés ou un liste de sélection à choix multiple. Rien ne change côté PHP, on récupère de la même manière nos informations, il faut juste bien garder en tête que nous traitons alors un tableau.

HTML

```
<form action='traitement.php' method='get'>
	<input type='checkbox' name='option[]' id='option1' value='1'>Rouge
	<input type='checkbox' name='option[]' id='option2' value='2'>Vert
	<input type='checkbox' name='option[]' id='option3' value='3'>Bleu
	<input type='submit' value='ENVOI'>
</form>option
```

PHP

```
<?php
   $options = $_GET['option'];
   print_r($options);
?>
```

PHP - Initiation

Formulaires et transmission de données

VIII.2.5) Vérifications et validations

Il peut arriver dans certains cas que le script PHP tente de récupérer la valeur d'un champ qui n'a pas été transmis (notamment dans le cas d'une **checkbox** non cochée), ce qui va générer une erreur. Afin d'éviter cela, il est d'usage de vérifier l'existence d'une variable avant récupération grace à la fonction **isset()**

```
var_dump(isset($test));          // va afficher : bool(false)
$test = 'ok';
var_dump(isset($test));          // va afficher : bool(true)
```

En appliquant cela à un traitement de case à cocher :

```
if (isset($_GET['accord'])) {
    $accord = 'oui';
} else {
    $accord = 'non';
}
```

PHP - Initiation

Formulaires et transmission de données

Attention : Les **select** en mode 'sélection multiple', les **checkbox** et les **radio** ne renvoient rien si l'utilisateur n'a fait aucun choix ou que les cases sont décochées. Les tests avec **isset()** deviennent primordiaux !

empty(*variable*)

Cette fonction alternative permet de vérifier non seulement l'existence d'une variable, comme **isset()**, mais également le fait qu'elle ne soit pas vide.

Les vérifications **javascript** côté client pouvant être désactivées ou carrément modifiées, elles ne suffisent pas à assurer la récupération correcte des données souhaitées. Tous les tests devront être doublés côté serveur, en **PHP**. (taille minimum ou maximum d'un texte, type de donnée, plage correcte, validité d'une adresse email, respect d'un format particulier...)

PHP - Initiation

Formulaires et transmission de données

VIII.3) Transmission de fichiers

VIII.3.1) Envoi (formulaire HTML)

Il nous faut avant tout modifier l'encodage dans la balise **form** avec l'attribut **enctype** :

<form action='*fichier*' method='post' enctype='multipart/form-data'>

Il nous reste ensuite à créer un **input** de type **file** pour permettre à l'utilisateur de choisir un fichier :

<input type='file' name='*monfichier*'>

Il est bien évidemment possible d'envoyer plusieurs fichiers d'un coup.
ATTENTION : Le temps d'attente de l'upload va s'allonger...

PHP - Initiation

Formulaires et transmission de données

VIII.3.2) Réception (PHP)

Lorsque le formulaire est soumis, le fichier est téléchargé sur le serveur hébergeant le formulaire, dans un dossier temporaire.
Il ne se passe rien d'autre : A nous de choisir ce qu'on fait de ce fichier.

Nous allons utiliser la super variable **$_FILES** pour récupérer toutes les informations utiles sur les fichiers qui viennent d'être téléchargés :

$_FILES['*monfichier*']['tmp_name']
Le nom et l'emplacement temporaire du fichier

$_FILES['*monfichier*']['name']
Le nom original du fichier

$_FILES['*monfichier*']['type']
Son type MIME (image/gif, application/pdf...)

PHP - Initiation

Formulaires et transmission de données

$_FILES['*monfichier*']['size']
Sa taille en octets (1 000 000 octets = 1Mo pour faire simple)

$_FILES['*monfichier*']['error']
Contient 0 (zéro) si tout s'est bien passé sinon un code d'erreur.

VIII.3.3) Traitement (PHP)

Tout comme pour les données plus classiques reçues d'un formulaire, il nous faut procéder à toute une batterie de test avant de valider le fichier.

Le fichier existe t-il ?

```
if (isset($_FILES['monfichier'])) {
    // Ici on continu...
}
```

PHP - Initiation

Formulaires et transmission de données

Le téléchargement s'est-il bien passé ?

```
if ($_FILES['monfichier']['error'] == 0) {
    // Ici on continu...
}
```

La taille ne dépasse t'elle pas la limite fixée ?

```
if ($_FILES['monfichier']['size'] < 2000000) {
    // Ici on continu...
}
```

Le type du fichier est-il celui attendu ?

```
if ($_FILES['monfichier']['type'] == 'image/png') {
    // Ici on continu...
}
```

ATTENTION : le type se base juste sur le nom de l'extension, on peut facilement tricher en la changeant.

PHP - Initiation

Formulaires et transmission de données

Lorsque tout est bon, on va accepter le fichier et le copier où bon nous semblera grace à une fonction dédiée :

move_uploaded_file(chemin_temporaire, chemin_final)

Lorsqu'on défini le chemin final pour notre fichier, nous pouvons reprendre son nom original mais rien ne nous empêche de le renommer.

Exemples :

```
move_uploaded_file($_FILES['monfichier']['tmp_name'], 'download/fichier.txt');
```

```
move_uploaded_file($_FILES['monfichier']['tmp_name'], 'download/'.$_FILES['monfichier']['name']);
```

PHP - Initiation

Formulaires et transmission de données

VIII.4) Sécurité

Les formulaires font partie des points les plus vulnérables des applications web. Si certaines failles sont facilement évitées, d'autres peuvent poser plus de problèmes et demander un attention particulière. Les informations que nous collectons provenant de l'extérieur, nous devons absolument les vérifier avant de les valider.

VIII.4.1) Défenses au niveau du HTML

L'utilisateur pouvant voir le code et même le copier pour l'utiliser depuis une de ses pages, tout ce que nous essaierons de faire dans notre formulaire HTML ne peut être considéré comme sûr.

Utiliser l'attribut **max_file_size** pour limiter la taille d'un fichier à envoyer peut être contourné, tout comme un **input** de type **hidden** peut être lu.

La seule chose primordiale à éviter est l'utilisation de la méthode **get** si on veut garder un minimum de confidentialité des données.
Mais là encore, des logiciels permettant la lecture des entêtes HTTP d'une requête en **post** sont malheureusement à la portée de n'importe qui aujourd'hui.

VIII.4.2) Défenses au niveau du script

Comme pour le HTML, les scripts étant côté client ils pourront facilement être contournés ou supprimés. Ils doivent être présent pour le confort de l'utilisateur et pour pallier aux erreurs ou tentatives malveillantes naïves.

PHP - Initiation

Formulaires et transmission de données

VIII.4.3) Défenses au niveau du PHP

Ce n'est véritablement qu'ici que nous allons pouvoir agir efficacement contre les tentatives mal intentionnées et les erreurs.

Tester l'existence

```
if (isset($_POST['prenom'])) {
    // Cette variable existe
} else {
    // Cette variable n'existe pas
}
```

Tester une valeur non vide

```
if ($_POST['prenom']!='') {
    // Cette variable n'est pas vide
} else {
    // Cette variable est vide
}
```

PHP - Initiation

Formulaires et transmission de données

Tester le type

```
if (is_string($_POST['prenom'])) {
    // Cette variable est bien une chaîne  de caractère
} else {
    // Cette variable n'est pas une chaîne de caractère
}
```

Tester le type va être très important surtout si l'on veut stocker ces informations dans une base de donnée. On a des équivalents pour tous les types :

- **is_int()** entier
- **is_float()** nombre décimal
- **is_numeric** entier ou décimal
- **is_bool()** booléen
- **is_array()** tableau

...

PHP - Initiation

Formulaires et transmission de données

Tester la taille

Pas assez utilisé dans les vérifications de données entrante, tester la taille d'une variable est pourtant simple et efficace. Par exemple refuser les prénoms qui auront plus de 25 caractères.

```
if (mb_strlen($_POST['prenom'])>25) {
    // Cette variable est trop longue
} else {
    // Cette variable à une taille correcte
}
```

Valeurs par défaut

Se débrouiller pour toujours avoir une valeur est une bonne pratique.

```
if ((isset($_POST['prenom']) && ($_POST['prenom']!='')) {
    // On récupère la valeur passée
} else {
    // On attribue une valeur par défaut, par exemple 'inconnu'
}
```

PHP - Initiation

Formulaires et transmission de données

Tester avec des REGEXP

Les expressions régulières sont parfaites pour vérifier les informations qui doivent respecter un format bien précis exigé.

```
$regexp = "/^[^0-9][A-z0-9_]+([.][A-z0-9_]+)*[@][A-z0-9_]+([.][A-z0-9_]+)*[.][A-z]{2,4}$/"

if (preg_match($regexp, $_POST['email']) {
    // Email valide
} else {
    // Email non valide
}
```

Tester avec filter_var

Extension désormais native depuis PHP 5.2, permet de faire de nombreuses vérifications bien pratiques !

```
if (filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    // Email valide
} else {
    // Email non valide
}
```

PHP - Initiation

Formulaires et transmission de données

Contrer les injections

Dans un champs texte par exemple, au lieu de simplement placer une chaine de caractère inoffensive, nous allons placer ce code :

```
'><script>alert('BOUM !')</script>
```

Ou celui-ci :

```
'><a href='http://www.google.fr'>mon site</a><img src='pirate.gif' title='ah ah ah
```

En utilisant un code HTML/PHP classique nous constatons rapidement qu'il y a un problème :

```
<input type='text' name='prenom' value='<?php echo $prenom; ?>'>
```

En effet, tous les caractères ont été insérés et certains sont interprétés par le HTML, changeant par là même le code d'origine. Pour éviter celà, nous devons transformer tous les caractères spéciaux HTML des informations que nous recevons.

PHP - Initiation

Formulaires et transmission de données

htmlspecialchars(chaîne, options)

Cette fonction renvoi la chaîne passée en encodant en HTML tout ce qui peux poser problème. Parmis les nombreuses options, deux nous intéressent particulièrement :

ENT_COMPAT Convertit les guillemets doubles, ignore les simples
ENT_QUOTES Convertit les guillemets doubles et simples

```
$test = "'><script>alert('BOUM !')</script>";
echo htmlspecialchars($test, ENT_QUOTES);

// Va afficher : &#039;&gt;&lt;script&gt;alert(&#039;BOUM !&#039;)&lt;/script&gt;
```