PHP

February 6, 2024

Les tableaux

Généralités

- tableaux scalaires :
  - les éléments sont indexés par des entiers
  - le 1er indice commence à 0
- tableaux associatifs :
  - les éléments sont indexés par des clés
  - clés = entier ou chaîne de caractères (pour un même tableau)
  - si la clé est une chaîne correspondant à un entier valide, la clé sera convertie en entier
    exemple : "2"
- les tableaux peuvent contenir des éléments de différents types
- Pas de taille à prédéfinir

Généralités

- tableaux scalaires :
  - les éléments sont indexés par des entiers
  - le 1er indice commence à 0
- tableaux associatifs :
  - les éléments sont indexés par des clés
  - clés = entier ou chaîne de caractères (pour un même tableau)
  - si la clé est une chaîne correspondant à un entier valide, la clé sera convertie en entier
    exemple : "2"
- les tableaux peuvent contenir des éléments de différents types
- Pas de taille à prédéfinir

Généralités

- tableaux scalaires :
  - les éléments sont indexés par des entiers
  - le 1er indice commence à 0
- tableaux associatifs :
  - les éléments sont indexés par des clés
  - clés = entier ou chaîne de caractères (pour un même tableau)
  - si la clé est une chaîne correspondant à un entier valide, la clé sera convertie en entier
    exemple : "2"
- les tableaux peuvent contenir des éléments de différents types
- Pas de taille à prédéfinir

Déclaration

```php
<?php
// Déclaration d'un tableau vide
$fruits = array();
// Déclaration d'un tableau indexé numériquement
$legumes = array('carotte','poivron','chou');
// Déclaration d'un tableau associatif
$identite = array(
'nom' => 'Hamon',
'prenom' => 'Hugo',
'age' => 19,
'estEtudiant' => true
);
?>
```

Ajout d'éléments

En spécifiant l'indice

```
$couleur[0] = 'bleu';
$couleur[2] = 'jaune';
$couleur[4] = 'orange';
```

Ajout d'un élément à la fin du tableau

```
$couleur[] = 'cyan';
```

- si tableau scalaire : indice = dernier indice +1
- si tableau associatif : indice = dernier indice numérique +1

Les tableaux multidimensionnels

= tableau de tableaux

```php
$tab = array(
0 => array("prenom"=>"pe", "nom"=>"wood","age"=>12),
1 => array("prenom"=>"fc", "nom"=>"bosque","age"=>11)
);
// accès
echo $tab[1]['prenom'] ; // affiche fc
```

Boucle

- Tableau scalaire

```php
$tab = array ("a", "b", "c", "d");
foreach ($tab as $val) {
print("$val<br/>");
}
```

- Tableau associatif

```php
$tab = array("prenom" => "paul", "nom" => "dupont");
foreach ($tab as $cle => $valeur) {
print("$cle = $valeur<br/>");
}
```

cf parcours du résultat de la requête SQL

Utilisation du pointeur

- Un tableau dispose d'un pointeur interne
- Par défaut il pointe sur le 1er élément
- il peut être déplacé à l'aide des fonctions

reset()	Place le pointeur au début du tableau et retourne la valeur du premier élément
next()	Déplace le pointeur sur l'élément suivant et retourne sa valeur
prev()	Déplace le pointeur sur l'élément précédent et retourne sa valeur
key()	Retourne la clé de l'élément courant
each()	Retourne un tableau contenant la paire clé/valeur de l'élément courant
current()	Retourne la valeur de l'élément courant
end()	Place le pointeur sur le dernier élément de la liste et retourne sa valeur

Utilisation du pointeur

Exemple :

```php
$tab = array("rouge","vert","bleu","jaune");
echo end($tab)."<br/>"; // affiche jaune
echo prev($tab)."<br/>"; // affiche bleu
echo key($tab)."<br/>"; // affiche 2
echo reset($tab)."<br/>"; // affiche rouge
echo current($tab)."<br/>"; // affiche rouge
```

Quelques fonctions utiles

- taille d'un tableau
  count($tab)
- Conversion chaînes / tableaux

```php
$liste = "patrick,pascal,veronique,benedicte,gonzague
    ,olivier";
$tab = explode(",",$liste);
```

```php
$date = "2004/10/23";
$tab = explode("/", $date);
```

Quelques fonctions utiles

Affectation d'un tableau à des variables : fct list()

```php
$info = array('coffee', 'brown', 'caffeine');
list($drink, $color, $power) = $info;
```

Combinaison explode() et list() : Conversion et affectation à des variables

```php
$date = "2004/10/23";
list($annee, $mois, $jour) = explode("/", $date);
```

Fonctions de "type" gestion de pile

$elem=array_shift($tab)	Retourne le premier élément du tableau et le supprime
$elem=array_pop($tab)	Retourne le dernier élément du tableau et le supprime
array_unshift($tab,'coucou',...)	Ajoute un ou plusieurs éléments au début du tableau
array_push($tab,'coucou',...)	Ajoute un ou plusieurs éléments à la fin du tableau

Fonctions de tri

sort($tab)	trie le tableau en fonction de ses valeurs
	ATTENTION si c'est un tableau associatif les clés sont supprimées
	elles sont remplacées par des indices numériques correspondant au nouvel ordre
asort($tab)	tri le tableau en fonction de ses valeurs
	les associations clé/valeur sont conservées
ksort($tab)	tri le tableau en fonction des clés
natsort($tab)	tri les valeurs du tableau avec l'ordre naturel
	img1.png < img2.png < img10.png < img12.png
	les associations clé/valeur sont conservées

Tri décroissant pour la fct sort = rsort()

Options de tri

Paramètres optionnels supplémentaires aux fonctions de tri

SORT_REGULAR	compare les éléments normalement (ne modifie pas les types)
SORT_NUMERIC	compare les éléments numériquement
SORT_STRING	compare les éléments comme des chaînes de caractères
SORT_NATURAL	compare les éléments comme des chaînes de caractères en utilisant l'ordre naturel comme le fait la fonction natsort().
SORT_FLAG_CASE	peut être combiné (grâce à l'opérateur OR) avec SORT_STRING ou SORT_NATURAL pour trier les chaînes sans tenir compte de la casse.

voir exemple `https://localhost/L3DAW/2023/cm5/Tableau_tri.php`

Présence d'un élément dans un tableau

Présence d'une valeur :

```php
$tab = array("thomas","henry");
if (in_array("thomas",$tab)) {
print("Le prénom Thomas a été trouvé");
}
```

Présence d'une clé :

```php
$tab = array("prenom"=>"thomas","nom"=>"henry");
if (array_key_exists("nom",$tab)) {
print("Le nom est défini");}
```

Présence d'un élément dans un tableau

Présence d'une valeur :

```php
$tab = array("thomas","henry");
if (in_array("thomas",$tab)) {
print("Le prénom Thomas a été trouvé");
}
```

Présence d'une clé :

```php
$tab = array("prenom"=>"thomas","nom"=>"henry");
if (array_key_exists("nom",$tab)) {
print("Le nom est défini");}
```

Sérialisation

= transformation d'une représentation binaire à une représentation textuelle.

```php
$tableau = array("couleur" => array("rouge", "jaune"),
1 => 2)
$tableau_s = serialize($tableau);
echo $tableau_s
// sotie  a:2:{s:7:"couleur";a:2:{i:0;s:5:"rouge";i:1;s
   :5:"jaune";}i:1;i:2;}
// a -> array i -> integer s -> string
```

UnSérialisation :

```php
$str = 'a:2:{i:0;s:1:"a";i:1;s:1:"b";}';
$tableau = unserialize($str);
```

Sérialisation

= transformation d'une représentation binaire à une représentation textuelle.

```php
$tableau = array("couleur" => array("rouge", "jaune"),
1 => 2)
$tableau_s = serialize($tableau);
echo $tableau_s
// sotie  a:2:{s:7:"couleur";a:2:{i:0;s:5:"rouge";i:1;s
   :5:"jaune";}i:1;i:2;}
// a -> array i -> integer s -> string
```

UnSérialisation :

```php
$str = 'a:2:{i:0;s:1:"a";i:1;s:1:"b";}';
$tableau = unserialize($str);
```

Opérateurs

$a + $b	Union de $a et de $b
$a == $b	Renvoie true si $a et $b sont composés des mêmes paires clé/valeur ; les tableaux sont alors dits égaux
$a === $b	Comme == , avec des vérifications supplémentaires sur l'ordre et le type des données; les tableaux sont alors dits identiques
$a != $b	Renvoie true si $a et $b ne sont pas égaux
$a !== $b	Renvoie true si $a et $b ne sont pas identiques

Date et heures

Notion de timestamp

Timestamp = horodateur

timestamp

- = nb de secondes écoulées depuis le 1er janvier 1970.
- **time()** retourne le timestamp actuel
- **date()** retourne une chaîne formatée de la date actuelle

Il faut manipuler les timestamp avec des fonctions

Fonctions pour timestamp

Créer un timestamp

- à partir des valeurs heure,minutes,...

```
$heure=0  ;$minute=9;$seconde=0;$mois=1;$jour=12;$anné
    e=2008;
mktime($heure,$minute,$seconde,$mois,$jour,$année);
```

- à partir d'une chaîne

```
strtotime("2008/01/12");
$time=strtotime("2008/01/12 00:01:00");
$time=strtotime("2008/01/12");
$time=strtotime("16 march 2008")
```

Fonctions pour timestamp

strtotime() "essaye" de lire une date au format anglais fournie en paramètre et la transforme en timestamp

```php
$time_semaine_suivante = strtotime("+7 days");
$time_semaine_suivante = strtotime("+1 week");
$time_semaine_suivante = strtotime("+6 days 24 hours");
```

Formatage d'une date

Formatage à l'aide de la fonction date(format[,timestamp])

date("d/m/y") :	02/02/15
date("d/m/Y") :	02/02/2015
date("d") :	02
date("l") :	Monday
date("l jS \of F Y h:i:s A") :	Monday 2nd of February 2015 03:36:34 PM

Options de format pour les jours

d	Jour du mois, sur deux chiffres (avec un zéro initial) 01 à 31
D	Jour de la semaine, en trois lettres (et en anglais - par défaut : en anglais, ou sinon, dans la langue locale du serveur) Mon à Sun
j	Jour du mois sans les zéros initiaux 1 à 31
l ('L' minuscule)	Jour de la semaine, textuel, version longue, en anglais Sunday à Saturday
N	Représentation numérique ISO-8601 du jour de la semaine (ajouté en PHP 5.1.0) 1 (pour Lundi) à 7 (pour Dimanche)
S	Suffixe ordinal d'un nombre pour le jour du mois, en anglais, sur deux lettres st, nd, rd ou th. Fonctionne bien avec j
w	Jour de la semaine au format numérique 0 (pour dimanche) à 6 (pour samedi)
z	Jour de l'année 0 à 365 Semaine — —
W	Numéro de semaine dans l'année ISO-8601, les semaines commencent le lundi (ajouté en PHP 4.1.0) Exemple : 42 (la 42ème semaine de l'année)

autres options de format

Pour les mois :

F	Mois, textuel, version longue; en anglais, comme January ou December January à December
m	Mois au format numérique, avec zéros initiaux 01 à 12
M	Mois, en trois lettres, en anglais Jan à Dec
n	Mois sans les zéros initiaux 1 à 12
t	Nombre de jours dans le mois 28 à 31

Pour les années :

L	Est ce que l'année est bissextile 1 si bissextile, 0 sinon.
o	L'année ISO-8601. C'est la même valeur que Y, excepté si le numéro de la semaine ISO (W) appartient à l'année précédente ou suivante, cette année sera utilisé à la place. (ajouté en PHP 5.1.0) Exemples : 1999 ou 2003
Y	Année sur 4 chiffres Exemples : 1999 ou 2003
y	Année sur 2 chiffres Exemples : 99 ou 03

Formatage en Français

Formate une date/heure locale avec la configuration locale

- définir la localisation :
  - setlocale() : Modifie les informations de localisation

```php
setlocale (LC_TIME, 'fr_FR.utf8','fra');
```

- strftime()

```php
$chaine_date_Fr=strftime("%A %d %B %Y.");
```

  - %A : Nom complet du jour de la semaine
  - %d : Jour du mois en numérique (01 à 31)
  - %B : Nom complet du mois, suivant la locale
  - %Y : L'année, sur 4 chiffres

Gestion des fichiers

Gestion des fichiers

Base de donnée vs gestion de fichiers

BD

- + fiabilité
- + représentation de structures complexes
- - complexe à mettre en œuvre
- - consommateur de ressources

Fichiers

- + simple à mettre en oeuvre
- + plus "performant"
- - il faut tout gérer (recherches, modifications, contrôle d'intégrité)

Optimisation

Remarques

- la diminution du nombre de requêtes SQL est de loin ce qui peut améliorer le plus les performances de vos scripts.
- Les optimisations sur les boucles et les diverses finesses du langage sont, en comparaison, négligeables.

Il peut être intéressant de mettre en place un système de cache pour les données :

- qui ne sont pas modifiée souvent
- qui sont consultées souvent

$\Rightarrow$ utilisation des fichiers

Optimisation

Remarques

- la diminution du nombre de requêtes SQL est de loin ce qui peut améliorer le plus les performances de vos scripts.
- Les optimisations sur les boucles et les diverses finesses du langage sont, en comparaison, négligeables.

Il peut être intéressant de mettre en place un système de cache pour les données :

- qui ne sont pas modifiée souvent
- qui sont consultées souvent

$\Rightarrow$ utilisation des fichiers

Ouverture, lecture, fermeture d'un fichier

fopen,fread,fclose

```
$descripeur_fichier = fopen("eleves.cache", "w+");
fread($descripeur_fichier, $taille_en_octet_de donné
    es_lues"));
fclose($descripeur_fichier);
```

Exemple ;

```
$fichier = fopen ("eleves.cache", "r");
$eleves = fread ($fichier, filesize("eleves.cache"));
// on lit la totalité du fichier
fclose($fichier);
echo $eleves;
```

Ouverture, lecture, fermeture d'un fichier

fopen,fread,fclose

```
$descripeur_fichier = fopen("eleves.cache", "w+");
fread($descripeur_fichier, $taille_en_octet_de donné
    es_lues"));
fclose($descripeur_fichier);
```

Exemple ;

```
$fichier = fopen ("eleves.cache", "r");
$eleves = fread ($fichier, filesize("eleves.cache"));
// on lit la totalité du fichier
fclose($fichier);
echo $eleves;
```

Les options

r	Ouverture en lecture seule (read)
w	Ouverture en écriture seule (write) ; le fichier est créé s'il n'existe pas
r+	Ouverture en lecture/écriture
w+	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, il est préalablement mis à zéro
a	Ouverture en écriture (append) ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier
a+	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier

Autres fonctions

fread	lit un nombre d'octets spécifié
fgets	récupère la ligne courante sur laquelle se trouve le pointeur du fichier
fwrite	écrit le contenu de la chaîne passée en paramètre
fputs	alias de fwrite
fseek	Modifie la position du pointeur de fichier
rewind	Replace le pointeur de fichier au début du flux
feof	Teste la fin du fichier

Seul le nom des fonctions est présenté, pour les paramètres voir la documentation.

Autres fonctions utiles

file_exists	Vérifie si un fichier ou un dossier existe.
is_writable	true si le fichier existe et est accessible en écriture
rename	Tente de renommer oldname en newname, en le déplaçant de répertoire si nécessaire.
disk_free_space	Renvoie l'espace disque disponible dans le répertoire ou la partition.

Seul le nom des fonctions est présenté, pour les paramètres voir la documentation.

Fichiers distants

Il est possible d'ouvrir un fichier distant : nom fichier = URL

```php
<?php
$fichier = fopen ("http://localhost/eleves.cache", "r");
$eleves = "";
while ($str = fread($fichier,16))
{
$eleves .= $str;
}
fclose($fichier);
echo $eleves;
?>
```

Remarque

- filesize() ne peut pas être utilisé pour les fichiers distants

Fichiers distants

via http en lecture seule mais via ftp en écriture (si autorisé):

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile
    ", "w");
if (!$file) {
  echo "<p>Impossible d'ouvrir le fichier distant pour é
      criture.\n";
  exit;
}
/* Ecriture des données. */
fputs ($file, $data_a_enregistrer . "\n");
fclose ($file);
?>
```

S'il est nécessaire de s'identifier, il existe des fonctions : ftp_connect(),...

les formulaires

Rappel sur les formulaires

Formulaire=

- Page web avec des zones de saisie (contrôles / widgets)
- L'utilisateur envoie au serveur les informations saisies en cliquant sur le bouton de type "submit".
- Chaque contrôle possède un attribut "name" utilisé pour transmettre l'information
- les données sont traitées par le script référencé initialement dans le formulaire

Rappel sur les formulaires

Formulaire=

- Page web avec des zones de saisie (contrôles / widgets)
- L'utilisateur envoie au serveur les informations saisies en cliquant sur le bouton de type "submit".
- Chaque contrôle possède un attribut "name" utilisé pour transmettre l'information
- les données sont traitées par le script référencé initialement dans le formulaire

Rappel sur les formulaires

Formulaire=

- Page web avec des zones de saisie (contrôles / widgets)
- L'utilisateur envoie au serveur les informations saisies en cliquant sur le bouton de type "submit".
- Chaque contrôle possède un attribut "name" utilisé pour transmettre l'information
- les données sont traitées par le script référencé initialement dans le formulaire

Rappel sur les formulaires

Formulaire=

- Page web avec des zones de saisie (contrôles / widgets)
- L'utilisateur envoie au serveur les informations saisies en cliquant sur le bouton de type "submit".
- Chaque contrôle possède un attribut "name" utilisé pour transmettre l'information
- les données sont traitées par le script référencé initialement dans le formulaire

Rappel sur les formulaires

```
<html>
<body>
 <h1>formulaire 1</h1>
  <form>
   <label>nom :</label> <input type="text" /><br/>
   <label>prenom :<label> <input type="text" /><br/>
   <input type="submit" />
  </form>
  <hr/>
  <h1>formulaire 2</h1>
  <form>
   <label>email :<label> <input type="text" /><br/>
   <input type="submit" />
  </form>
</body>
</html>
```

Sélection du script

- l'attribut "action" de la balise "form" spécifie le script qui sera appelé pour traiter le formulaire
- exemple

```
<form action="script.php">
```

- s'il n'y a pas d'attribut action le formulaire est envoyé à lui-même
- On peut référencer un script sur un autre serveur (url)

```
<form action="http://www.google.fr/search">
<input type="text" name="q" />
<input type="submit" value="nouvelle recherche" />
</form>
```

Les méthodes d'envoi des données

Il y a 2 méthodes, GET et POST, pour envoyer les données du formulaire.

La méthode GET (méthode par défaut)

- transmet LES DONNÉES via l'URL
- sous forme d'une chaîne de paramètres (Query String)
- le nom du script est suivi de '?' puis de la Query String
- format de la Query String:
  nom1=valeur1&nom2=valeur2&...&nomN=valeurN

Les méthodes d'envoi des données

Il y a 2 méthodes, GET et POST, pour envoyer les données du formulaire.

La méthode GET (méthode par défaut)

- transmet LES DONNÉES via l'URL
- sous forme d'une chaîne de paramètres (Query String)
- le nom du script est suivi de '?' puis de la Query String
- format de la Query String:
  nom1=valeur1&nom2=valeur2&...&nomN=valeurN

Les méthodes d'envoi des données

Il y a 2 méthodes, GET et POST, pour envoyer les données du formulaire.

La méthode GET (méthode par défaut)

- transmet LES DONNÉES via l'URL
- sous forme d'une chaîne de paramètres (Query String)
- le nom du script est suivi de '?' puis de la Query String
- format de la Query String:
  nom1=valeur1&nom2=valeur2&...&nomN=valeurN

Les méthodes d'envoi des données

Il y a 2 méthodes, GET et POST, pour envoyer les données du formulaire.

La méthode GET (méthode par défaut)

- transmet LES DONNÉES via l'URL
- sous forme d'une chaîne de paramètres (Query String)
- le nom du script est suivi de '?' puis de la Query String
- format de la Query String:
  nom1=valeur1&nom2=valeur2&...&nomN=valeurN

Récupération des données

- dans le script php on utilise la variable superglobale $_GET
- Exemple pour le formulaire

```
<form action="script.php">
<label>nom :</label> <input name="nom" type="text" />
    <br/>
<label>prenom :<label> <input name="prenom" type="
    text" /><br/>
<input type="submit" />
</form>
```

- Script de traitement (script.php):

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

Récupération des données

- dans le script php on utilise la variable superglobale $_GET
- Exemple pour le formulaire

```
<form action="script.php">
<label>nom :</label> <input name="nom" type="text" />
    <br/>
<label>prenom :<label> <input name="prenom" type="
    text" /><br/>
<input type="submit" />
</form>
```

- Script de traitement (`script.php`):

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

Les différents types de contrôles

Les contrôles à valeur unique :

- zone de texte normal

```
<input type="text" name="nom" />
```

- Pour mot de passe

```
<input type="password" name="motdepasse" />
```

- zone de texte multiligne

```
<textarea name="desciption"></texarea>
```

menu déroulant à choix unique

```
<select name="test">
<option value="1"> premier choix </option>
<option value="2"> deuxième choix </option>
</select>
```

Les différents types de contrôles :

Les contrôles à valeur multiple :
Menu à choix multiple

```
<select name="test[]" multiple="true" size="3">
<option value="1"> premier choix </option>
<option value="2"> deuxième choix </option>
<option value="3"> troisième choix </option>
</select>
```

Dans le formulaire:

- Il faut donner comme attribut "name" un nom suivi de '[]'

Dans le script php:

- $_GET['test'] est un tableau contenant les options choisies
- si "name" sans '[]', on ne récupère que la dernière valeur sélectionnée

Les différents types de contrôles :

Les contrôles à valeur multiple :
Menu à choix multiple

```
<select name="test[]" multiple="true" size="3">
<option value="1"> premier choix </option>
<option value="2"> deuxième choix </option>
<option value="3"> troisième choix </option>
</select>
```

Dans le formulaire:

- Il faut donner comme attribut "name" un nom suivi de '[]'

Dans le script php:

- $_GET['test'] est un tableau contenant les options choisies
- si "name" sans '[]', on ne récupère que la dernière valeur sélectionnée

Checkbox

```
choix1<input type="checkbox" name="choix[]" value="val1"
    />
choix2<input type="checkbox" name="choix[]" value="val2"
    />
choix3<input type="checkbox" name="choix[]" value="val3"
    />
```

Dans le script php:

- idem menu à choix multiple
- mais on peut donner des noms différents pour chaque checkbox

Bouton radio

```
choix 1 <input type="radio" name="choix" value="radio1" /
    >
choix 2 <input type="radio" name="choix" value="radio2" /
    >
choix 3 <input type="radio" name="choix" value="radio3" /
    >
```

Dans le script php le tableau n'est pas nécessaire comme une seule valeur est retournée.

La méthode POST

- Les données du formulaire sont transmises de manière cachée (dans le corps de la requête HTTP).
- On spécifie la méthode dans le formulaire : attribut 'method'

```
<form action="auth.php" method="post">
  <label>identifiant</label>
  <input type="text" name="id" /><br/>
  <label>mot de passe</label>
  <input type="password" name="pass" /><br/>
  <br/><input type="submit" value="identification">
</form>
```

La méthode POST

- Les données du formulaire sont transmises de manière cachée (dans le corps de la requête HTTP).
- On spécifie la méthode dans le formulaire : attribut 'method'

```
<form action="auth.php" method="post">
  <label>identifiant</label>
  <input type="text" name="id" /><br/>
  <label>mot de passe</label>
  <input type="password" name="pass" /><br/>
  <br/><input type="submit" value="identification">
</form>
```

La méthode POST

- Dans le script l'accès aux données se fait de façon analogue à la méthode GET via la variable superglobale $_POST.

```php
<?php
print("<b>Identifiant</b> : ". $_POST['id']."<br/>");
print("<b>Mot de passe</b> : ".$_POST['pass']."<br/>"
    );
?>
```

▸ Voir exemple via LiveHTTPHeaders

GET vs POST

	GET	POST
Taille des données	limitée	illimitée
Visibilité de données	visible dans l'URL	cachées dans le corps de la requête HTTP mais en clair

Remarque :

variable superglobale $_REQUEST permet d'accéder au contenu de$_GET et $_POST

GET vs POST

	GET	POST
Taille des données	limitée	illimitée
Visibilité de données	visible dans l'URL	cachées dans le corps de la requête HTTP mais en clair

Remarque :

variable superglobale $_REQUEST permet d'accéder au contenu de$_GET et $_POST

Traitement des formulaires

- Production du formulaire
- Validation de la saisie
- Récupération des données saisies
- Contrôle des données saisies
- Enregistrement des données

Production du formulaire

"À la main" en html

- si fixe et ne dépend pas de données
  Exemple : formulaire d'identification (identifiant/ mot de passe)
- doit être simple et les modifications peu fréquentes

Automatiquement à partir de données

- provenant d'une base de données
  $\Rightarrow$ nécessite l'écriture d'un programme (PHP) pour lire les données et produire le formulaire.
- provenant d'un fichier XML
  - $\Rightarrow$ le formulaire peut être généré par une feuille de style XSL
  - $\Rightarrow$ on peut écrire un programme (PHP) pour interpréter le fichier XML et produire le formulaire.
- Exemple : QCM

Production du formulaire

"À la main" en html

- si fixe et ne dépend pas de données
  Exemple : formulaire d'identification (identifiant/ mot de passe)
- doit être simple et les modifications peu fréquentes

Automatiquement à partir de données

- provenant d'une base de données
  ⇒ nécessite l'écriture d'un programme (PHP) pour lire les données et produire le formulaire.
- provenant d'un fichier XML
  - ⇒ le formulaire peut être généré par une feuille de style XSL
  - ⇒ on peut écrire un programme (PHP) pour interpréter le fichier XML et produire le formulaire.
- Exemple : QCM

Production du formulaire

"À la main" en html

- si fixe et ne dépend pas de données
  Exemple : formulaire d'identification (identifiant/ mot de passe)
- doit être simple et les modifications peu fréquentes

Automatiquement à partir de données

- provenant d'une base de données
  $\Rightarrow$ nécessite l'écriture d'un programme (PHP) pour lire les données et produire le formulaire.
- provenant d'un fichier XML
  - $\Rightarrow$ le formulaire peut être généré par une feuille de style XSL
  - $\Rightarrow$ on peut écrire un programme (PHP) pour interpréter le fichier XML et produire le formulaire.
- Exemple : QCM

Production du formulaire

"À la main" en html

- si fixe et ne dépend pas de données
  Exemple : formulaire d'identification (identifiant/ mot de passe)
- doit être simple et les modifications peu fréquentes

Automatiquement à partir de données

- provenant d'une base de données
  $\Rightarrow$ nécessite l'écriture d'un programme (PHP) pour lire les données et produire le formulaire.
- provenant d'un fichier XML
  - $\Rightarrow$ le formulaire peut être généré par une feuille de style XSL
  - $\Rightarrow$ on peut écrire un programme (PHP) pour interpréter le fichier XML et produire le formulaire.
- Exemple : QCM

Validation de la saisie avant envoi

Pourquoi?

- Vérifier si les données saisies sont conformes
  - Au format attendu (ex :N de téléphone)
  - Aux valeurs attendues
- Éviter d'envoyer des données qui ne peuvent être traitées
- Augmenter l'interactivité/réactivité de la page

Validation de la saisie avant envoi

Pourquoi?

- Vérifier si les données saisies sont conformes
  - Au format attendu (ex :N de téléphone)
  - Aux valeurs attendues
- Éviter d'envoyer des données qui ne peuvent être traitées
- Augmenter l'interactivité/réactivité de la page

Validation de la saisie avant envoi

Pourquoi?

- Vérifier si les données saisies sont conformes
  - Au format attendu (ex :N de téléphone)
  - Aux valeurs attendues
- Éviter d'envoyer des données qui ne peuvent être traitées
- Augmenter l'interactivité/réactivité de la page

Validation de la saisie avant envoi

Pourquoi?

- Vérifier si les données saisies sont conformes
  - Au format attendu (ex :N de téléphone)
  - Aux valeurs attendues
- Éviter d'envoyer des données qui ne peuvent être traitées
- Augmenter l'interactivité/réactivité de la page

Validation de la saisie avant envoi

Comment?

- Traité en amont à la production du formulaire
  - liste de choix
  - contrôle spécifique <input type="tel" (html5)
- Traité en directe
  - Avec une fonction Javascript
  - Attachée à chaque contrôle (onclick/onchange/onfocus/onkeydown/onkeypress/onkeyup,...)
  - et analysant le contenu ou les frappes clavier
  - Attaché au formulaire

```
<form .... onSubmit="return verifform()">
```

Validation de la saisie avant envoi

Comment?

- Traité en amont à la production du formulaire
  - liste de choix
  - contrôle spécifique <input type="tel" (html5)
- Traité en directe
  - Avec une fonction Javascript
  - Attachée à chaque contrôle (onclick/onchange/onfocus/onkeydown/onkeypress/onkeyup,...)
  - et analysant le contenu ou les frappes clavier
  - Attaché au formulaire

```
<form .... onSubmit="return verifform()">
```

Récupération des données saisies

cf diapos précédentes

Conseil

- Privilégiez la variable SUPERGLOABLE $_REQUEST (plutôt que $_GET ou $_POST
- Évite de devoir réécrire les scripts PHP si on change la méthode d'envoi du formulaire

Contrôle de la saisie du formulaire

Pourquoi, si on a déjà fait un contrôle de la saisie?

- La validation avant envoi ne peut pas tout garantir
  - peut nécessiter des données non-transmises
    - Données confidentielles (ex : mot de passe)
    - Données trop volumineuses
  - Traitement de validation trop complexe
  - Traitement de validation confidentiel
- Intervention de l'internaute
  - Exécution JavaScript Désactivé
  - Injection de code / requêtage http directe,...

Contrôle de la saisie du formulaire

Pourquoi, si on a déjà fait un contrôle de la saisie?

- La validation avant envoi ne peut pas tout garantir
  - peut nécessiter des données non-transmises
    - Données confidentielles (ex : mot de passe)
    - Données trop volumineuses
  - Traitement de validation trop complexe
  - Traitement de validation confidentiel
- Intervention de l'internaute
  - Exécution JavaScript Désactivé
  - Injection de code / requêtage http directe,...

Contrôle de la saisie du formulaire

Pourquoi, si on a déjà fait un contrôle de la saisie?

- La validation avant envoi ne peut pas tout garantir
  - peut nécessiter des données non-transmises
    - Données confidentielles (ex : mot de passe)
    - Données trop volumineuses
  - Traitement de validation trop complexe
  - Traitement de validation confidentiel
- Intervention de l'internaute
  - Exécution JavaScript Désactivé
  - Injection de code / requêtage http directe,...

Contrôle de la saisie du formulaire

Pourquoi, si on a déjà fait un contrôle de la saisie?

- La validation avant envoi ne peut pas tout garantir
  - peut nécessiter des données non-transmises
    - Données confidentielles (ex : mot de passe)
    - Données trop volumineuses
  - Traitement de validation trop complexe
  - Traitement de validation confidentiel
- Intervention de l'internaute
  - Exécution JavaScript Désactivé
  - Injection de code / requêtage http directe,...

Contrôle de la saisie du formulaire

Pourquoi, si on a déjà fait un contrôle de la saisie?

- La validation avant envoi ne peut pas tout garantir
  - peut nécessiter des données non-transmises
    - Données confidentielles (ex : mot de passe)
    - Données trop volumineuses
  - Traitement de validation trop complexe
  - Traitement de validation confidentiel
- Intervention de l'internaute
  - Exécution JavaScript Désactivé
  - Injection de code / requêtage http directe,...

Contrôle de la saisie du formulaire

Comment?

- Vérifications du contenu (isempty()/isset())
- Analyser le contenu des variables à l'aide des fonctions de traitement de chaînes
  - Nettoyage : trim(), stript_tag, htmlspecialchars()... ou filter_var() et filtre de nettoyage
  - Analyse : expressions régulières ou filter_var() et filtre de validation
- Traitements : ils dépendent de l'application

Contrôle de la saisie du formulaire

Comment?

- Vérifications du contenu (isempty()/isset())
- Analyser le contenu des variables à l'aide des fonctions de traitement de chaînes
  - Nettoyage : trim(), stript_tag, htmlspecialchars()... ou filter_var() et filtre de nettoyage
  - Analyse : expressions régulières ou filter_var() et filtre de validation
- Traitements : ils dépendent de l'application

Contrôle de la saisie du formulaire

Comment?

- Vérifications du contenu (isempty()/isset())
- Analyser le contenu des variables à l'aide des fonctions de traitement de chaînes
  - Nettoyage : trim(), stript_tag, htmlspecialchars()... ou filter_var() et filtre de nettoyage
  - Analyse : expressions régulières ou filter_var() et filtre de validation
- Traitements : ils dépendent de l'application

Contrôle de la saisie du formulaire

Comment?

- Vérifications du contenu (isempty()/isset())
- Analyser le contenu des variables à l'aide des fonctions de traitement de chaînes
  - Nettoyage : trim(), stript_tag, htmlspecialchars()... ou filter_var() et filtre de nettoyage
  - Analyse : expressions régulières ou filter_var() et filtre de validation
- Traitements : ils dépendent de l'application

Enregistrement des données

Module de gestion des données

- Accès à la structure de données
- Assurer le contrôle d'intégrité des données si on n'utilise pas une base de données
- Gestion des conflits/doublons (insertion / modification)