Le traitement des données de formulaire

Le langage PHP de base

El hadji Mamadou NGUER Enseignant chercheur en Informatique à l'UVS

Séquence 5 : Le traitement des données de formulaire

Objectifs spécifiques : A la suite de cette séquence, l'étudiant doit être capable de :

1. Décrire les variables d'environnement du langage PHP
2. Traiter les données de formulaire dans une page PHP.
3. Créer et mettre en ligne un site web PHP.

Séquence 5 : Le traitement des données de formulaire

Plan de la séquence :

1. Introduction
2. Les variables d'environnement
3. Traitement des données de formulaires
4. Validation des données de formulaire
5. Conclusion

Introduction

- Les actions les plus communes dans un site web sont :

  - la connexion,
  - l'affichage de données,
  - l'ajout de données,
  - la modification de données,
  - la suppression de données,
  - la déconnexion.

- Et ces actions passent par des formulaires dont le traitement des données nécessitent l'utilisation de variables d'environnement.

- Nous allons dans cette séquence voir comment utiliser les variables d'environnement pour le traitement des formulaires et leur validation.

Les variables d'environnement

- Les variables d'environnement ont été introduites en PHP 4.1.0.
- Ce sont des tableaux associatifs toujours accessibles, indépendamment de la portée.
- En effet, on peut y accéder à partir de n'importe quelle fonction, classe ou fichier sans avoir à faire quelque chose de spécial. Elles sont :

$GLOBALS
$_SERVER
$_FILES
$_ENV
$_COOKIE
$_SESSION
$_REQUEST
$_POST
$_GET

- Quelques unes sont expliquées dans cette séquence. Le reste sera vu ultérieurement.

Les variables d'environnement

$GLOBALS

- Elle est utilisée pour accéder aux variables globales de n'importe où dans un script PHP (aussi de l'intérieur des fonctions ou méthodes).
- PHP stocke toutes les variables globales dans un tableau appelé $GLOBALS [*index*]. *index* contient le nom de la variable.

Exemple

```
<?php
  $x = 75;
  $y = 25;
  function addition() {
          $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
  }

  addition();
  echo $z;
?>
```

Les variables d'environnement

$_SERVER

- Elle contient des informations sur les en-têtes, les dossiers et les chemins du script.

Exemple

```
<?php
   echo $_SERVER['PHP_SELF'];    // nom du script en cours
   echo "<br>";
   echo $_SERVER['SERVER_NAME'];
   echo "<br>";
   echo $_SERVER['HTTP_HOST'];  echo "<br>";   // adresse IP du client
   echo $_SERVER['HTTP_REFERER'];
   echo "<br>";
   echo $_SERVER['SCRIPT_NAME'];
?>
```

Les variables d'environnement

- **$_ENV :** Elle contient des informations liées à l'environnement dans lequel s'exécute le script.

```
<?php  echo $_ENV["USER"]. " exécute actuellement ce script"; ?>
```

- **$_FILES :** contient des fichiers uploadés via un formulaire.

- **$_SESSION :** contient des données de session. Il est utilisé pour la connexion et déconnexion d'un site web.

- **$_COOKIE:** contient des données de cookie. Il est aussi utilisé pour la connexion et déconnexion d'un site web.

- **$_REQUEST, $_POST et $_GET :** seront vues dans le traitement des données de formulaire.

Traitement des formulaires

- Les variables globales $_REQUEST, $_GET et $_POST sont utilisées pour recueillir les données envoyés depuis un formulaire ou depuis un lien ($_GET).
- Soit le formulaire ci-dessous.

```
<html>
<body>

<form action="nomfichier.php" method="post">
        Login: <input type="text" name="login"><br>
        Mot de passe: <input type="text" name="mdp"><br>
        <input type="Soumettre">
</form>

</body>
</html>
```

Fichier qui traite les données

Méthode d'envoi des données.
Valeurs possibles : POST ou GET

Données à envoyer

Bouton d'envoi des données

Traitement des formulaires

- L'exemple ci-dessus montre un formulaire de méthode d'envoi post avec deux champs de saisie (de nom login et mdp) et un bouton d'envoi.

- Lorsqu'un utilisateur envoie les données en cliquant sur «**Soumettre**», les données de formulaire sont envoyées au fichier spécifié dans l'attribut action de la balise form.

- Dans cet exemple, on a mis « nomfichier.php » pour traiter les données mais on peut aussi mettre le même fichier contenant le formulaire.

```
action="<?php echo $_SERVER['PHP_SELF'];?>"
```

- Nous allons maintenant voir comment utiliser les variables $_REQUEST, $_POST et $_GET pour traiter les données du formulaires.

Traitement des formulaires

$_REQUEST

- Elle est utilisée pour collecter des données après la soumission d'un formulaire HTML de méthode d'envoi get ou post.
- **Exemple**

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
        Name: <input type="text" name="fname">
        <input type="submit">
</form>

<?php
        $name = $_REQUEST['fname'];
        echo $name;
?>
</body>
</html>
```

Traitement des formulaires

$_POST

- Elle est largement utilisée pour collecter des données de formulaire soumis avec method = "post".

- **Exemple**

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
        Name: <input type="text" name="fname">
        <input type="submit">
</form>
<?php
        $name = $_POST['fname'];
        echo $name;
?>
</body>
</html>
```

Traitement des formulaires

$_GET

- Elle peut également être utilisée pour collecter des données de formulaire soumis avec method = "get".
- $ _GET peut également collecter des données envoyées dans l'URL.

  Supposons que nous avons une page HTML qui contient un lien hypertexte avec des paramètres:

```
<html>
<body>

<a href="test_get.php?sujet=PHP&web=ufrsat.sn">Test $GET</a>

</body>
</html>
```

Traitement des formulaires

$_GET

- Lorsqu'un utilisateur clique sur le lien "Test $_GET", les paramètres "sujet" et "web" sont envoyés à "test_get.php", et vous pouvez alors recueillir leurs valeurs dans "test_get.php" avec $_GET.
- L'exemple ci-dessous montre le code dans "test_get.php":

Exemple

```
<html>
<body>

<?php
echo "Etudier" . $_GET['sujet'] . " à " . $_GET['web'];
?>

</body>
</html>
```

Traitement des formulaires

Le code précédent est assez simple. Cependant, il manque la chose la plus importante. Vous devez valider les données de formulaire pour protéger votre script contre un code malveillant.

Pensez à la SÉCURITÉ lors du traitement des formulaires PHP!

Cette page ne contient pas de validation de formulaire, il montre comment vous pouvez envoyer et récupérer des données de formulaire.

Dans la suite, nous allons montrer comment traiter les formulaires PHP en y mettant de la sécurité! Une validation des données du formulaire est importante pour se protéger contre des hackers et les spammeurs!

Traitement des formulaires

GET vs POST

• Les méthodes GET et POST créent un tableau (par exemple **array (key1 => valeur, key2 => valeur2, key3 => valeur3, ...)**). Ce tableau contient des paires clé / valeur, correspondant aux noms des champs du formulaires et aux valeurs respectivement saisies par l'utilisateur.

• Les deux GET et POST sont traités avec $_GET et $_POST.

• $_GET est un tableau de variables passées au script via les paramètres d'URL.

• $_POST est un tableau de variables passées au script via la méthode HTTP POST.

Traitement des formulaires

Quand utiliser GET?

• Les informations envoyées à partir d'un formulaire, avec la méthode GET, sont **visible à tout le monde** (tous les noms et les valeurs des variables sont affichées dans l'URL).

• GET a également des limites sur le nombre d'informations à envoyer. La limitation est d'environ 2000 caractères.

• Cependant, parce que les variables sont affichées dans l'URL, il est possible d'ajouter la page. Ceci peut être utile dans certains cas.

• GET peut être utilisée pour envoyer des données non sensibles.

Remarque: GET doit jamais être utilisé pour l'envoi de mots de passe ou d'autres informations sensibles!

Traitement des formulaires

Quand utiliser POST?

• Les informations envoyées à partir d'un formulaire, avec la méthode POST, sont **invisible** (les noms et les valeurs sont intégrés dans le corps de la requête HTTP) et ne présente de **limite** sur la quantité d'informations à envoyer.

• De plus POST supporte des fonctionnalités avancées telles que les entrées binaires multi-part lors du l'envoi de fichiers vers le serveur.

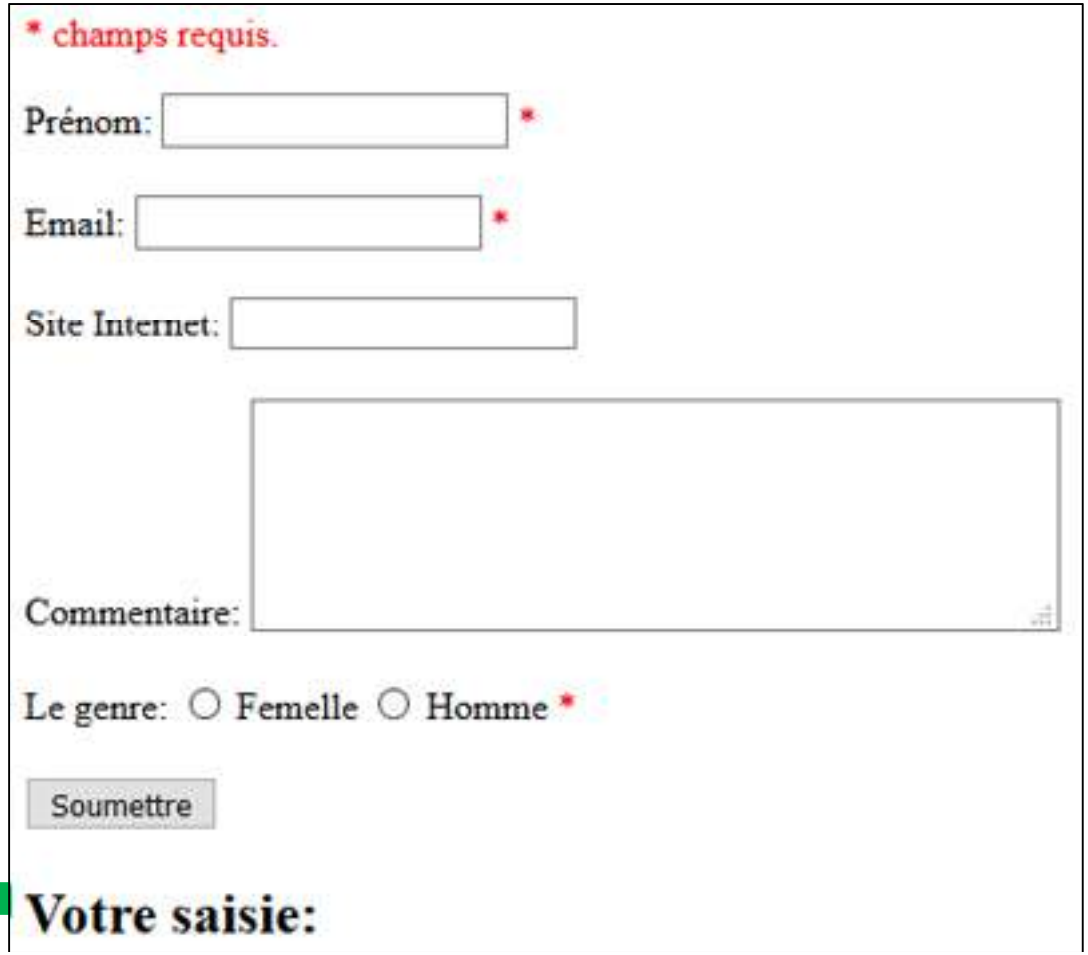
• Cependant, parce que les variables ne sont pas affichés dans l'URL, il n'est pas possible de mettre en signet la page.

• **Les développeurs préfèrent POST pour l'envoi des données de formulaire.**

Validation des données de formulaire

Cette partie montrent comment utiliser PHP pour valider les données de formulaire c.à.d. traiter les formulaires en toute sécurité.

- **Exemple de formulaire pour validation**



Validation des données de formulaire

Composition du formulaire

Le formulaire est composé de :

- **champs de texte** dont le code ressemble à ceci :

Nom: <input type="text" name="nom">
E-mail: <input type="text" name="email">
Site web: <input type="text" name="website">
Commentaire: <textarea name="commentaire" rows="5" cols="40"></textarea>

- **Boutons de radio** dont le code ressemble à ceci :

Genre:
<input type="radio" name="genre" value="feminin">Feminin
<input type="radio" name="genre" value="masculin">Masculin

- **L'élément de formulaire**

Le code HTML du formulaire ressemble à ceci:
<form method="post" action="<?php echo **htmlspecialchars**($_SERVER["PHP_SELF"]);?>">

Validation des données de formulaire

- **Règles de validation**

Les règles de validation pour le formulaire ci-dessus sont les suivantes:

Champs	Règles de validation
Nom	Champs obligatoires + Ne doit contenir que des lettres et des espaces
E-mail	Champs obligatoires + Doit contenir une adresse électronique valide (avec @ et.)
Site web	Optionnel. Si présent, il doit contenir une URL valide
Commentaire	Optionnel. Champ de saisie multi-lignes (zone de texte)
Genre	Champs obligatoires. Doit en choisir un.

Validation des données de formulaire

Composition du formulaire

- **L'élément de formulaire**

Lorsque le formulaire est soumis, les données du formulaire sont envoyées avec method = "post".

$ _SERVER ["PHP_SELF"] est une variable super globale qui renvoie le nom du fichier du script en cours d'exécution.

Ainsi, $ _SERVER ["PHP_SELF"] envoie les données du formulaire soumises à la page elle-même, au lieu de sauter sur une page différente. De cette façon, l'utilisateur recevra des messages d'erreur sur la même page que le formulaire.

La fonction **htmlspecialchars ()** convertit des caractères spéciaux en entités HTML. Cela signifie qu'il remplacera des caractères HTML tels que < et > par < et >. Cela empêche aux hackers d'exploiter le code en injectant du code HTML ou Javascript (Cross-site Scripting Attack) dans les formulaires.

Validation des données de formulaire

Notes sur la sécurité des formulaires

- La variable $ _SERVER ["PHP_SELF"] peut être utilisée par les pirates informatiques! Si PHP_SELF est utilisé dans votre page, un utilisateur peut entrer une barre oblique (/), puis des commandes de **scripts inter-sites (**Cross Site Scripting (XSS)) à exécuter.
- **Les scripts inter-sites (XSS) sont un type de vulnérabilité de sécurité informatique généralement trouvée dans les applications Web. XSS permet aux hackers d'injecter des scripts côté client dans des pages Web vues par d'autres utilisateurs.**
- **Exemple:** Supposons avoir le formulaire suivant dans une page "test_form.php":

  <form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
- Maintenant, si un utilisateur entre l'URL normale dans la barre d'adresse comme "http://www.example.com/test_form.php", le code ci-dessus sera traduit en:

  <form method="post" action="test_form.php">

Ce qui marche très bien.

Validation des données de formulaire

Notes sur la sécurité des formulaires

- Cependant, considérez qu'un utilisateur entre l'URL suivante dans la barre d'adresse:

http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E

Dans ce cas, le code ci-dessus sera traduit en:
<form method="post" action="test_form.php/"><script>alert('hacked')</script>

- Ce code ajoute une balise script et une commande d'alerte. Et lorsque la page est chargée, le code JavaScript sera exécuté (l'utilisateur verra une fenêtre d'alerte). Ce n'est qu'un exemple simple et inoffensif de la manière dont la variable PHP_SELF peut être exploitée.

- Soyez conscient que **tout code JavaScript peut être ajouté dans la balise <script>!** Un pirate peut rediriger l'utilisateur vers un fichier sur un autre serveur et ce fichier peut contenir un code malveillant pouvant modifier les variables globales ou soumettre le formulaire à une autre adresse pour enregistrer les données utilisateur, par exemple.

Validation des données de formulaire

Comment éviter l'exploitation de $ _SERVER ["PHP_SELF"]?

- L'exploitation de $_SERVER ["PHP_SELF"] par les hackers peut être évité en utilisant la fonction **htmlspecialchars ()**.

- Le code de formulaire devrait ressembler à ceci**:<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">**

- La fonction htmlspecialchars () convertit les caractères spéciaux en entités HTML de telle sorte que si l'utilisateur tente d'exploiter la variable PHP_SELF, il en résultera la sortie suivante:
- <form method="post" action="test_form.php/"><script>alert('hacked')</script>">

- La tentative d'exploitation échoue et aucun mal n'est causé!

Validation des données de formulaire

Les étapes de validation des données de formulaire avec PHP

Pour chaque variable de données de formulaire :

- Etape 1: Passer la variable via la fonction htmlspecialchars () de PHP
- Etape 2: Dégager les caractères inutiles (espace supplémentaire, onglet, nouvelle ligne) à partir de la variable de données de formulaire (avec la fonction PHP trim ())
- Etape 3 : Supprimer les antislash (\) des données d'entrée de l'utilisateur (avec la fonction stripslashes () de PHP)
  - Cependant il est plus pratique que ces trois étapes soient effectués dans une fonction qui fera toutes les vérifications pour chaque variable.
  - Ces étapes de validation de données de formulaire sont mis en œuvre dans l'exemple suivant.

Validation des données de formulaire

Les étapes de validation des données de formulaire avec PHP

- **Exemple:**

```php
<?php
// définir et initialiser les variables
$nom = $email = $genre = $commentaire = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["nom"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["commentaire"]);
  $gender = test_input($_POST["genre"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
```

Validation des données de formulaire

Les étapes de validation des données de formulaire avec PHP

Remarques :

- Notez qu'au début du script, on a vérifié si le formulaire a été envoyé avec $_SERVER["REQUEST_METHOD"].

  ✓ Au cas échéant, le formulaire est soumis et validé.

  ✓ Sinon on ignore la validation et un formulaire vierge est affiché.

- Notons aussi que, dans l'exemple ci-dessus, tous les champs de saisie sont facultatifs. Le script fonctionne bien même si l'utilisateur n'entre pas de données.

- L'étape suivante consiste à créer les champs de saisie obligatoire ainsi que des messages d'erreur si nécessaire.

Validation des données de formulaire

Gestion des champs obligatoires avec PHP

- Ce chapitre montre comment créer des champs de saisie obligatoires ainsi que des messages d'erreur si besoin.

- Nous avions dans le diapo 94 fixé que les champs nom, email et genre sont obligatoires. Ainsi ils ne doivent pas être vides.

- Dans le code PHP, la fonction **empty()** sera utilisée pour vérifier si les variables de données de formulaire correspondantes au champs ci-mentionnés sont vides ou pas. Des messages d'erreur seront alors générés au cas contraire :

Exemple :

```
if (empty($_POST["nom"])) {
  $errNom = "Le nom est obligatoire";
} else {
  $nom = test_input($_POST["nom"]);
}
```

Affichage du message d'erreur:

```
Nom: <input type="text" name="name">
<span class="error">*
<?php echo $errNom;?></span>
<br><br>
```

Validation des données de formulaire

D'autres validations

- **Validation du nom**

Le code ci-dessous montre un moyen simple de vérifier si le champ nom contient uniquement des lettres et des espaces.

```
$nom = test_input($_POST["nom"]);
if (!preg_match("/^[a-zA-Z ]*$/",$nom)) {
  $errNom = "Seules les lettres et les espaces blancs sont autorisés";
}
```

La fonction preg_match () recherche un motif dans une chaîne. Elle renvoie vrai si le motif existe et false sinon.

Validation des données de formulaire

D'autres validations

- **Validation d'e-mail**

Le moyen le plus simple et le plus sûr de vérifier si une adresse e-mail est bien formée est d'utiliser la fonction **filter_var ()** de PHP.

Exemple:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Format d'email invalide";
}
```

- **Validation d'URL**

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "URL invalide";
}
```

Mise en ligne d'un site web

Les motivations

- Durant ce cours nous avons appris à créer un site web dynamique en local
- Cependant un site web créé en local n'est pas accessible sur le web;
- Pour qu'il soit acessible sur le Web, il faudra :

1. réserver un **nom de domaine**
2. trouver un **hébergeur** pour votre site web
3. importer éventuellement la base de données locale vers la base de données de l'hébergeur
4. enfin, utiliser un **client FTP** pour transférer le site vers le serveur

Mise en ligne d'un site web

Le nom de domaine

- Un **nom de domaine** est un identifiant de domaine internet.
- Un **domaine internet** est un ensemble d'ordinateurs reliés à Internet et possédant une caractéristique commune.

Exemple : un domaine tel que .sn est l'ensemble des ordinateurs hébergeant des activités pour des personnes ou organisations qui se sont enregistrées auprès de Nic Sénégal (nic.sn) qui est le registre responsable de premier niveau du domaine .sn.

En général, ces personnes ou ces entreprises ont en général une certaine relation avec le Sénégal. Par exemple le domaine gouv.sn est l'ensemble des ordinateurs hébergeant des activités pour le gouvernement du Sénégal.

Mise en ligne d'un site web

Le nom de domaine

- Un nom de domaine est un « masque » sur une adresse IP. Son but est de retenir et communiquer facilement l'adresse d'un ensemble de serveurs (site web, courrier électronique, FTP).

Par exemple, **gouv.sn** est plus simple à mémoriser que 196.1.94.186.

Syntaxe

- Un nom de domaine est constitué de deux parties : ***nomdomaine.extension***
  - où ***nomdomaine*** est le nom du domaine proprement dit
  - ***extension*** est l'extension (aussi appelée « TLD », de l'anglais *top-level domain*). Il existe grosso modo une extension par pays.

  - En général, un site web voit son adresse précédée par www, comme par exemple www.gouv.sn. www ne fait pas parti du domaine, c'est juste un sous domaine.

Mise en ligne d'un site web

Le nom de domaine

Pour réserver un nom de domaine, on a deux solutions :

- Passer par un **registrar** spécialisé. C'est un organisme qui sert d'intermédiaire entre l'ICANN (l'organisation qui gère l'ensemble des noms de domaine au niveau international) et vous. Kheweul.com SA et Sonatel Multimedia sont de célèbres registrars sénégalais.

- Vous pouvez aussi commander le nom de domaine en même temps que l'hébergement (c'est ce qui est conseillé).

- Cependant on peut avoir un **nom de domaine gratuit** si on décide d'héberger gratuitement son site web.

Mise en ligne d'un site web

L'hébergement d'un site web

- Sur Internet, tous les sites web sont stockés sur des serveurs (figure ci-dessus.

- La société mettant à votre disposition un serveur web connecté en permanence à internet est appelée hébergeur.

- On distingue deux principales catégories d'hébergeurs :

  - les hébergeurs gratuits (000webhost.com par exemple).

  - les hébergeurs professionnels (arvixe.com par exemple).

- L'hébergeur vous fournit le **nom du serveur**, les **paramètres de connexion** nécessaire pour transférer votre site ainsi qu'un **cPanel** qui permet plusieurs paramétrages dont celui du serveur de base de données.

Mise en ligne d'un site web

Le transfert d'un site web

- Le transfert du site web vers le serveur se fait à l'aide d'un client FTP comme Filezilla. Mais il faudra disposer des paramètres de connexion au serveur fournis par l'hébergeur.

Mise en ligne d'un site web

L'accès au cPanel

- **cPanel** est un panneau de configuration basé sur Linux conçu pour les hébergeurs web
- L'accès au **serveur de base de données** peut se faire facilement via le cPanel

Conclusion

- Au cours de cette séquence, nous avons vu :
  - Les fonctions en PHP, précisément :
    - La description d'une fonction
    - La création d'une fonction
    - L'appel d'une fonction
    - Et les paramètres d'une fonction
  - Les tableaux, précisément :
    - tableaux indexés
    - Les tableaux associatifs
    - Et les tableaux multidimensionnels
- La séquence suivante sera consacrée au traitement des données de formulaire