Premières notions de PHP

(petit aide-mémoire, ceci n'est pas un cours)

- Le PHP est un langage interprété par le serveur HTTP, pour générer des pages HTML qui sont envoyées au client (navigateur).
- Une page PHP porte l'extension .php ; c'est un fichier texte.
- Le code PHP dans la page est délimité par

```
<?php
// ici votre code
?>
```

- Le PHP ressemble au C (et à la famille des langages issus du C, comme le C#). La plupart des instructions s'écrivent comme en C : if(…), while(…), for(…), switch(…)
  Les commentaires se notent comme en C :

```
// pour la fin de ligne
/*  pour
un commentaire
sur plusieurs lignes */
```

- Les noms de variables commencent toujours par $
- Les variables n'ont pas à être déclarées ; ceci prête parfois à confusion, car une variable « fausse » est acceptée, par exemple :

```
$prix = $prix * 2 ;
echo "Le prix est $prrix" ;
```

  Ceci affichera un prix vide car la « fausse » variable $prrix sera considérée comme valide mais vide.
- En revanche les tableaux doivent être déclarés :
  Exemple d'un tableau de participants :

```
$participants=array() ;
```

  Remarquons que la taille du tableau n'est pas définie. Elle sera gérée dynamiquement.
  Exemple : ajout d'un élément à un tableau

```
$participants[]='Vanessa' ;
```

  ajoute la participante 'Vanessa' en fin de tableau.
- Un tableau peut s'utiliser avec des indices de composantes, comme en C# :
  $participants[0]='Albert' ;
- Tableau associatif : On associe un nom à une composante, voir ci-dessous $_POST, $_GET, $_SESSION.
- Les variables sont locales, c'est-à-dire qu'elles appartiennent à une page ; lorsqu'on passe d'une page à une autre, les variables de « l'ancienne » page disparaissent. Idem pour les tableaux (d'où la nécessité de passer les valeurs des variables par GET, POST ou SESSION).

Pour inclure du HTML dans une page, deux solutions :
1. Le mettre directement en HTML, en dehors du code < ?php…. ?>
Exemple :

```
<h1>Description des produits</h1>
<?php
// ici par exemple une boucle pour afficher toutes les lignes d'un tableau
?>
<a href='index.php?action=accueil'>retour</a>
```

2. Utiliser l'instruction echo : (ici avec une variable $couleur)

```
echo "<tr><td>Couleur</td><td>$couleur</td></tr>" ;
```

Tableaux associatifs :
Un tableau associatif est un tableau qui ***associe*** un nom à chacune de ses composantes.

$_GET
Tableau associatif qui reçoit les données passées par URL
`http://www.monsite.fr/exemple4.php?nom=cow&prenom=flying`
crée automatiquement dans la page `exemple4.php`
`$_GET["nom"]` (qui contient la valeur "cow")
`$_GET["prenom"]` (qui contient la valeur "flying")

$_POST
Tableau associatif qui reçoit les données passées par POST (en général formulaire)
```
<form method='POST' action='index.php?action=controlesaisie'  …
…
<input type='text' name='nom' size='20'>
```
crée automatiquement dans la page `index.php`
`$_POST["nom"]` (qui contient la valeur saisie dans le champ de « name » nom)

$_REQUEST
Tableau associatif qui regroupe les données de $_GET, $_POST (et $_COOKIE), et l'identifiant de session (SID) suivant la configuration du serveur

$_SESSION
Tableau associatif qui stocke les données de session

- ➔ il est impératif que la page appelle la session, en première instruction dans la page :
  `session_start() ;`
- ➔ Pour stocker dans la session :
  `$_SESSION["login"]=$login ;`
- ➔ Pour lire une donnée stockée :
  `$nom=$_SESSION["nom"] ;`

Le tableau $_SESSION peut stocker des tableaux, mais dans ce cas il faut le déclarer :
```
// en supposant qu'on a un tableau déjà existant $les_participants
$_SESSION["participants"]=array() ;
$_SESSION["participants"]=$les_participants ;
```

Déterminer si une variable existe :
Il est parfois nécessaire de vérifier qu'une variable existe, par exemple dans la session.
Fonction « **`isset()`** » : est déterminée
```
if (isset($_SESSION["login"])
{
…
```

Il est possible de déterminer l'inverse en utilisant le « non » logique : « ! »
```
if ( ! isset($_SESSION["login"])
{
…
```

Autre possibilité : Fonction « empty() » : est vide

Insertion de code :
Lorsqu'on utilise souvent les mêmes paramètres ou les mêmes fonctions, on peut « factoriser » le code, c'est-à-dire le mettre dans un fichier à part qui sera appelé dans toutes les en-têtes
`include "paramconnexion.php" ;`
va charger le code contenu dans paramconnexion.php
Il est plus prudent d'utiliser :
`require "paramconnexion.php" ;`
qui rend le chargement du fichier obligatoire ; si le fichier est absent, l'exécution s'arrêtera.

Redirection :
Une page peut appeler une autre page de 3 façons :

- Par un lien « ancre » HTML
  `<a href='index.php?action=accueil'>retour</a>`
  Ici c'est le clic de l'utilisateur sur le lien qui déclenche le passage à l'URL `index.php` avec une variable `action` en GET

- Par un bouton « submit » dans un formulaire
  ```
  <form method='POST' action='index.php?action=controlesaisie'  …
  …
  <input type='submit' value='Ajouter'></td>
  </form>
  ```
  Le clic de l'utilisateur sur "Ajouter" appellera la page `index.php` en passant **deux** types d'éléments :
  une variable `action` dans le tableau `$_GET`
  les données des « `input` » dans le tableau `$_POST`

- Redirection interne au serveur :
  `header("Location: v_accueil.php")`
  Une page ***interne au serveur*** peut appeler une autre page, à condition qu'aucun caractère HTML n'ait été envoyé au client (et en particulier, attention aux espaces qui trainent… invisibles, ils sont néanmoins considérés comme un caractère HTML).
  Par exemple un contrôleur (souvent index.php) s'exécute sur le serveur sans rien afficher (donc sans rien envoyer au client), et peut donc appeler une autre page (vue ou modèle).