Mémento PHP orienté pratique ICT-133

Les bases du php, les spécialités et points importants.

Règles et notions générales:

- Pas besoin de déclarer les variables et leur type.
- Toutes les variables commencent par $. Ca indique que ce qui suit c'est une variable.
- Tout le code php se trouve dans une balise PHP:

  Syntaxe simplifiée de la balise php: On peut remplacer: `<?php echo $name ?>` par `<?= $name ?>` quand il n'y a qu'une valeur à afficher (plus besoin de `echo` du coup).

- Il y a des ";" à la fin des lignes !

  ```
  $x = 12;
  $y = "Hello";
  ```

  Attention comme il n'y a pas de déclaration du types de variables, de ne pas avoir deux variables différentes avec le même nom (pour deux valeurs différentes):

  ```
  $x = 12;
  $x = "Hello";
  ```

  Les variables peuvent **changer de type** au cours du code. On peut faire un changement implicite par un calcul:

  `$i = $_GET['month'];` $i est de type string:
  `$i +=0;` on fait un calcul et le type change.
  `switch($i){ ...` on peut donc l'utiliser comme un int
  puis on fait `$i .= ""` on le concatène avec rien (.= meme principe que +=) et il devient de type string.

Manipulation de chaines de caractères:

Avec l'acronyme **MERCI**:

- **Mesurer**: `count($tableau)` retourne nombres d'éléments du tableaux, ou `strlen($myrandomstring)` retourne la longueur de la chaine.
- **Extraire**: `substr(thestring, startpos, length)` retourne la chaine extraite
- **Rechercher**: `stripos ( string $haystack , mixed $needle [, int $offset = 0 ] ) : int` cherche une aiguille dans une botte de foin donc une sous chaine dans une chaine.
- **Concaténer**: `"<li>Produit ".$i."</li>"` On met un "." au lieu du "+" en C#.
- **Interpoler**: `"<li>Produit $i</li>"` mais double guillemets obligatoires. Attention, ça ne fonctionne pas avec les tableaux ! On utilise donc la concaténation pour les tableaux.

Beaucoup d'autres fonctions existent sur https://www.php.net :

- `str_replace("_", " ", $filename)` remplacer les "_" par des " " dans filename.

- `str_repeat ( string $input , int $multiplier ) : string` Repeat a string
- `substr_count()` --> Count the number of substring occurrences

Affichage de dates:

`date ( string $format [, int $timestamp = time() ] ) : string`
`$format` = format voulu ("Y-m-d" par exemple). `$timestamp` = par défaut le temps de maintenant ou une date donnée.
Toutes les syntaxes sont sur ce lien pour la fonction `date()`
https://www.php.net/manual/en/function.date.php
Pour se mettre sur le fuseau horaire de la suisse, on ajoute:
`date_default_timezone_set("Europe/Zurich");`

Exemples de syntaxes:

```
echo "<li>".date('l d F Y')."</li>";
echo "<li>".date('M jS Y')."</li>";
echo "<li>".date('d/m/Y H:i a')."</li>";
echo "<li>".date('d M Y, H:i:s')."</li>";
echo "<li>".date('r')."</li>";
```

Résultats:

Thursday 28 November 2019
Nov 28th 2019
28/11/2019 11:41 am
28 Nov 2019, 11:41:02
Thu, 28 Nov 2019 11:41:02 +0100

Principe du gabarit:

La gabarit c'est un modèle (concrètement une page html qui contient en-tête et pied de page) et qui contient des zones qui sont générées par d'autres pages. C'est un template.

MVC

MVC = Modèle, Vue et Contrôleur. Cela consiste à séparer les données, de ce qui est affiché et de la logique effectuée, en séparant les pages php qui ont un rôle bien précis. Détails dans ![MVC explication.md](MVC explication.md)

Lier les pages:

Puisque chaque pages à un but particulier (en MVC) mais qu'une seule page ne suffit pas, il faut pouvoir les lier.

3 manières de lier les pages entre elles:

- `require('nomfichier');`
- `require_once('nomfichier');`
- `include();`

Dans les 3 cas, le résultat est exactement le même que si on copiait à la place le contenu du fichier appelé.

Différences: si le fichier appelé n'existe pas:

- require: crash
- require_once: crash
- include: n'inclut rien.

  ATTENTION: Comme les pages php sont comme copiés collés dans le fichier qui faire require, le **chemin relatif** des fichiers sont relatifs **par rapport à la première page** (page demandée dans la requête).

  Toutes les variables en dehors de celles des fonctions sont accessibles directement par toutes les pages. Attention cependant à l'ordre dans lequel sont liés les pages et à l'initialisation des variables.

Buffer (mémoire tampon)

Utilité: Permettre de générer un certain contenu sans l'afficher directement mais en faisant comme si on l'affichait. Donc on fait des `echo` et des bouts de html seuls mais ca n'est pas affiché. Ca va dans le buffer.

Fonctions pour le buffer:

`ob_start();` output buffer start = met un charriot à la sortie de la salle. (tout ce qui va suivre ca va dans le buffer)
`$content = ob_get_clean();` on récupère le contenu du charriot et on le met dans $content

Exemple:

```
<?php
ob_start(); //départ du buffer
?>
<a href='?action=movies'><img src="images/movies.jpg" alt="movies"
height="250px"></a>
<a href='?action=concerts'><img src="images/concerts.jpg" alt="movies"
height="250px"></a>
<?php
$content = ob_get_clean();recevoir le buffer
?>
```

`isset($_GET['action']);` retourne si valeur est définie (false si nulle)
`unset();` vider/rendre nulle une variable

Boucle Foreach:

Littéralement "pour chaque" donc c'est une boucle for améliorée qui prend tous les éléments. Pas besoin de savoir combien il y en a donc. C'est différent qu'en C# et l'élément `$concert` n'est qu'une copie et pas un lien sur l'élément réel. C'est parfait pour accéder en lecture mais pas fait pour faire des modifications ou des suppressions.

```
foreach($listconcerts as $concert){
echo $concert['name'];
}
```

Pour contourner on peut prendre l'index de l'élément en cours et utiliser le tableau réel:

```
foreach($listofconcerts as $i => $concert){
unset(listofconcerts[$i]);
}
```

Tableaux:

Traditionnelement un tableau se crée avec la fonction array();

```
$contacts = array("John", "David", "Romain", "Jules");
```

La nouvelle syntaxe permet de remplacer `array( ... )` par `[ ... ]`.

3 types de tableaux:

Les tableaux indexés (comme en C)

Les valeurs sont numérotées avec un index partant de 0.

```
$cars = array("Volvo", "BMW", "Toyota");
```

- Volvo est à l'index 0
- BMW est à l'index 1
- ...

Pour utiliser tous les éléments il suffit de faire une boucle. On écrit ou lit la valeur avec le numéro d'index entre crochet. $cars[index]

Les tableaux associatifs

Chaque valeur est lié (`=>`) à une clé. La clé désigne la signification de la valeur.
`$contactInfo = array( 'name' => 'John Doe', 'address' => 'Rue de Lausanne 25', 'NPA' => 1400, 'City' => 'Yverdon' );`

- John Doe est lié à name
- 1400 est relié à NPA.

On ne peut pas faire de boucles FOR puisque il n'y a plus d'index, alors on fait une boucle foreach:

```
foreach ($contactInfo as $info){
echo $info." "; ici $info prend chaque valeur du tableau.
}
```

Si on oublie de mettre un clé, un index est automatiquement mis.

Ou alors pour prendre une case dont on connait la clé, on met la clé entre '[' et ']'
`$contactInfo["name"] = "John Assange";`

Les tableaux multidimensionnels:

```
$people = array(
array('Perceval','Arthur','Lancelot','Leodagan'),
array('Marge','Homer','Bart','Maggie'),
array('Joe','Jack','William','Averell')
);
```

ou

```
$people = [
['Perceval','Arthur','Lancelot','Leodagan'],
['Marge','Homer','Bart','Maggie'],
['Joe','Jack','William','Averell']
];
```

`$people` est ici un tableau indexés de tableaux indexés.

En créant des tableaux dans une case, on obtient un tableau 1D dans un case donc deux dimensions finalement. Il est possible d'avoir autant de dimensions que souhaités.

Fonctions pour tableau: extract();