TP 3 : Jukebox (statique) (avec solutions)

Objectifs du TP:

- Exercices sur la dynamicité du langage.
- Premier site WEB avec 2 pages.

1. Exercices : Opérateurs, Types et structures dynamiques

Lorsque l'on aborde un nouveau langage, il faut connaître sa catégorie : PHP est un langage impératif, procédural et orienté objet, avec un typage dynamique. Il faut aussi connaître les points de similarité avec les autres langages pour réduire le temps d'apprentissage, lorsque l'on connait déjà ces langages (cela devrait être le cas pour vous). PHP est inspiré du langage C pour les structures (if, for, while, switch, etc.), pour les appels de fonctions (passage par copie) et pour les librairies (les mêmes issues d'Unix). Il est inspiré de C++ (passage des paramètres par référence), et par Java pour son modèle objet. Il est finalement inspiré du shell pour les variables (le $) et le comportement des chaines de caractères (différence entre ' et "). Sachant cela, il ne reste alors plus qu'à se concentrer sur les quelques différences avec ces autres langages.

Dans les exercices suivants, nous allons nous concentrer sur le typage dynamique, la durée de vie des variables et leur portée (ou visibilité), la structure particulière des tableaux associatifs et la dynamicité des attributs d'instances du modèle objet.

1.1 Types dynamiques et comparaisons

Un langage avec typage dynamique comme PHP, dispense le programmeur de la déclaration du type des variables. Une variable PHP n'a donc pas de type : c'est la valeur qui lui est associée qui possède un type. On peut forcer l'évaluation d'une variable ou d'une expression, vers un autre type en utilisant des fonctions de transtypage.

Rappels du cours : la concaténation transforme toute valeur en chaîne, les opérations arithmetiques transforment toute valeur en nombre. La transformation d'une chaîne en nombre donne le nombre présent en début de chaine ou le nombre zéro. La comparaison (ex: "=="), avec un nombre, transforme une chaîne en nombre. L'opérateur "===" ne retrouve vrai que pour des valeurs identiques et de **même type**.

A faire :

1. Testez le code php suivant :

```
<?php
$a = 1;
$a .= '0';
$b = 2 * 5;
print('<p>$a est de type ' . gettype($a) . ' et a pour valeur "' . $a . '"</p>');
print('<p>$b est de type ' . gettype($b) . ' et a pour valeur ' . $b . '</p>');
if ($a == $b) {
    echo "<p>Avec l'opérateur == : '$a' et $b sont équivalents</p>";
}
if ($a === $b) {
    echo "<p>Avec l'opérateur === : '$a' et $b sont identiques</p>";
} else {
    echo "<p>Avec l'opérateur === : '$a' et $b sont différents</p>";
}
```

▼ Solution :
Le code affiche :

```
$a est de type string et a pour valeur "10"
$b est de type integer et a pour valeur 10
Avec l'opérateur == : '10' et 10 sont équivalents
Avec l'opérateur === : '10' et 10 sont différents
```

2. Examinez la page web http://php.net/manual/fr/language.operators.comparison.php. A partir des informations de cette page, donner une explication à ce résultat.

▼ Solution :
Comme l'opérateur de concaténation transforme tout en chaîne, l'entier 1 devient la chaine "1", concaténé à l'autre chaine, elle devient la chaine de valeur "10". Note : l'opérateur '.=' est un raccourcis de l'opération "$a = $a .". La variable $b contient l'entier

```
10
```

résultat de la multiplication.

L'opérateur '==' fait un test après transtypage, comme il y a au moins un nombre à tester, la chaine est transformée en entier. Comme la chaîne contient l'entier 10, le transtypage transforme cette chaîne en l'entier 10 qui est égal au contenu de $b.

Par contre comme les deux variables sont de types différents, le deuxième test retourne faux.

Conclusion : pour un code sécurisé qui ne teste que des valeurs de même type, préférer l'opérateur

```
===
```

.

1.2 Portée, durée de vie et visibilité des variables

Rappel du cours :

- La portée d'un variable initialisée dans un **contexte global** est la totalité du script. Par contre sa visibilité est limité aux expressions du contexte global, sauf si le mot clé 'global' est utilisé dans le bloc d'une fonction ou méthode. Sa durée de vie est toujours celle du script.
- La portée d'une variable initialisée dans le **bloc d'une fonction (ou méthode)** va de la ligne de son initialisation jusqu'à la fin du bloc. Sa visibilité est limité au bloc (donc pas visible dans les fonctions appelées). Sa durée de vie est celle du bloc. Le mot clé 'global' permet de rendre visible une variable initialisée dans un contexte global.
- Le mot clé 'static' **étend la durée de vie** d'une variable initialisée dans le bloc d'une fonction (ou méthode) à tout le script, mais ne change pas sa portée qui reste locale.

1. Testez le code php suivant :

```
function bonjour() {
  if (isset($nom)) {
    echo "Bonjour $nom</br>";
  } else {
    echo "Mais qui êtes vous ?</br>";
  }
}

function hello() {
  global $nom;
  if (isset($nom)) {
    echo "Hello $nom</br>";
  } else {
    echo "Mais qui êtes vous ?</br>";
  }
}

function salut() {
  static $nom;
  if (isset($nom)) {
    echo "Salut $nom</br>";
  } else {
    echo "Mais qui êtes vous ?</br>";
  }
  $nom = "Cyprien";
}

bonjour();
$nom="Arthur";
bonjour();

hello();
$nom="Marcel";
hello();

salut();
$nom="Mohamed";
salut();
```

2. Expliquez pourquoi on obtient l'affichage :

```
Mais qui êtes vous ?
Mais qui êtes vous ?
Hello Arthur
Hello Marcel
Mais qui êtes vous ?
Salut Cyprien
```

Aidez-vous des explications de la page : http://php.net/manual/fr/language.variables.scope.php

▼ Réponse :
- Pour bonjour(), la variable $nom est locale et n'est pas initialisée donc `isset($nom)` est toujours evalué à faux.
- Pour hello() le $nom est global et c'est la valeur de $nom global avec un changement de valeur qui est affiché.
- Pour salut() le nom est local mais rémanant c'est à dire qu'il garde son état même après la sortie de la fonction. Sa durée de vie est celle du script.

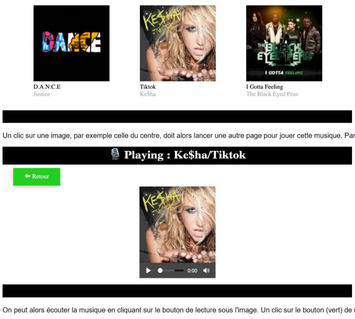
2. Un Jukebox dans un site WEB

Dans cette partie nous vous proposons de réaliser un jukebox, c'est-à-dire une page WEB qui affiche des images représentant des musiques et qui les fait jouer lorsque l'on clique dessus. Ce travail sera développé plusieurs TP. Dans cette première partie, il s'agit de faire un site statique, c'est à dire le contenu est pré-définie dans la page HTLM.

Dans les TP suivants, vous produirez un site dynamique, c'est à dire, qui affiche la liste des musiques qui sont décrites dans un fichier. L'étape suivante consistera à remplacer ce fichier par une base de données. Finalement, la dernière étape consistera à structurer votre code selon le patron de conception MVC.

2.1 Jukebox statique

Dans cette première version, l'affichage des musiques est placé dans une page statique qui s'affiche de la manière suivante :



Un clic sur une image, par exemple celle du centre, doit alors lancer une autre page pour jouer cette musique. Par exemple :

On peut alors écouter la musique en cliquant sur le bouton de lecture sous l'image. Un clic sur le bouton (vert) de retour doit faire revenir à la page principale.

La page principale `staticJukebox.html` est fournie ainsi que le fichier de style `style.css` dans un fichier zip à télécharger. Ce fichier contient aussi des données : des images au format jpeg et des musiques au format mp3, dans le sous répertoire `data`. Dans ce répertoire `data`, chaque sous répertoire est un groupe de musique. Il contient alors toutes les musiques de ce groupe sous la forme de deux fichiers :

- `XXX.mp3` : pour la musique.
- `XXX.jpeg` : pour l'illustation de la musique `XXX.mp3`.

Contenu de l'archive :

Dans cet exercice, on designe une musique à jouer par une chaîne de caractères qui comprend le nom du groupe, le caractère `/`, suivi du nom de la musique. Dans l'exemple précédent on a demandé à jouer `Ke$ha/Tiktok`. Pour obtenir une **URL relative** à partir de cette information, il suffit d'ajouter au début le chemin relatif vers les données : `data/` et à la fin le suffixe pour indiquer le type de fichier. Par exemple, à partir du répertoire où se trouve la page principale, l'URL relative vers la musique `Ke$ha/Tiktok` est `data/Ke$ha/Tiktok.mp3`

Travail à faire :

1. Recopiez dans votre répertoire de TP (sous public_html) tous les fichiers et répertoires contenus dans l'archive suivante : data/staticJukebox.zip
2. Récupérez les données de musiques dans l'archive suivante : data/dataJukebox.zip
3. Vérifiez que vous pouvez afficher cette première page avec votre *navigateur*, à travers une URL. Examiner le contenu de cette page statique : `staticJukebox.html`. Remarquez l'appel au fichier `playPath.php` dans un lien autour de l'image. Examinez également comment est réalisé le style de cette page.
4. Réalisez le programme `playPath.php` qui prend en paramètre dans l'identifiant `music` de la **query string** l'identification de la musique à jouer et de l'image à afficher. Pour lancer un player audio, utilisez la balise HTML5 :

```
<audio src="data/Justice/D.A.N.C.E.mp3" controls="" autoplay=""></audio>
```

▼ Solution :

Le fichier `playPath.php` :

```
<?php
// Vérification de la présence du paramètre
if (isset($_GET['music'])) {
  $music = $_GET['music'];
} else {
  /// En l'absence du paramètre prend une valeur par défaut
  $music = 'Justice/D.A.N.C.E';
}
// calcule les chemin des fichiers
$musicPath = 'data/'.$music;
$mp3 = $musicPath.'.mp3';
$cover = $musicPath.'.jpeg';
?>
<html>
<head>
  <meta charset="UTF-8">
  <title>Mon jukebox static: play</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <header>
    <h1>Playing : <?= $music ?></h1>
  </header>
  <main>
    <nav>
      <a href="staticJukebox.html">&#x2805; Retour</a>
    </nav>
    <section>
      <figure>
        <img src="<?= $cover ?>"/>
        <audio src="<?= $mp3 ?>" controls autoplay ></audio>
      </figure>
    </section>
  </main>
  <footer>
  </footer>
</body>
</html>
```

Récupération de la solution complète dans un fichier ZIP : data/JukeBoxStatic.zip

▼ La solution complète dans un seul fichier ZIP :

Fichier à télécharger : data/JukeboxStatic.zip