Exercice 3.3 : Le Morpion

Le but de ce TP est de faire un mini-jeu de morpion sur une page Web. Pour cela on va utiliser le HTML/CSS pour l'affichage et le PHP pour effectuer le traitement des données.

Le but final est d'avoir un rendu ressemblant à celui-ci :
https://rgrondin.perso.centrale-marseille.fr/morpion/

Compétences abordées : HTML/CSS, PHP, Formulaires, Sessions, Redirections, Données GET/POST, Algo (basique)

Prérequis

Il est nécessaire d'avoir suivi Devweb 101 : Les bases du développement web, et d'avoir les outils et compris les bases du PHP présentées sur Devweb 103 : Le back-end et PHP.

Cahier des Charges

Description de ce qui devra être fait

Le jeu consistera en une même page pouvant afficher différents contenu selon l'état d'avancement du joueur.

- Lors d'une première visite : une page demandant le nom des deux joueurs.
- Une fois ce formulaire validé : La page de jeu. Celle-ci doit comporter :
  - Un texte indiquant le tour du joueur
  - La grille de jeu, avec les croix (X), les ronds (O) sous forme d'un tableau, avec des liens cliquables pour jouer à un endroit.
  - Le nom des deux joueurs et à quel signe ils correspondent.
- Il faut détecter lorsqu'un joueur a gagné et lorsque la grille est remplie (match nul)
- Une fois une victoire ou un match nul détecté, il faut indiquer le joueur gagnant ou le match nul et proposer de rejouer. (Réinitialisation du jeu)

Variables du jeu

Le jeu devra comporter 4 variables pour enregistrer les données des joueurs :

```
$_SESSION['nom1'] = 'Nom du joueur 1';
$_SESSION['nom2'] = 'Nom du joueur 2';
$_SESSION['tour'] = 1; // Tour du joueur 1 ou 2
$_SESSION['grid'] = [ // Valeur 0 pour pas joué, 1 -> joueur 1 et 2 ->
joueur 2
    [0, 0, 0],
    [0, 0, 0],
```

```
    [0, 0, 0] ];
```

Le nom des variables avec `$_SESSION` sera expliqué plus tard

Joueur sur une case

Pour jouer sur une case, il faudra se rendre sur le lien : *morpion.php?**ligne=0**&**colonne=2*** pour jouer en (1,3). (On commence à 0 pour que ce soit plus simple avec les tableaux.)

Première partie : L'affichage

On commence avec un code basique disposant uniquement d'un peu de style et d'un titre. Ça sera notre base pour la suite.

morpion.php

```
<!DOCTYPE html>
<html>
    <head>
        <title>Morpion</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Morpion de l'ambiance</h1>
    </body>
</html>
```

style.css

```
body{
  background-color: darkgreen;
  color: white;
}
h1,h3{
  text-align: center;
}
```

Pour l'instant cette page dispose de peu de choses, juste d'un fond vert, de titre centrés (On utilise *text-align: center;* pour les titres h1 et h3 car il s'agit de balises de type block, qui prennent tout l'écran par défaut)

Gérer quelle page est affichée

Une petite astuce pour gérer l'affichage des pages simplement sans que le code soit un foutoir est de stocker dans une variable le nom du bloc à afficher, et de faire des conditions dans la partie HTML pour déterminer ce qu'il faut afficher. Un exemple :

Remarquez que lorsqu'on met du PHP, il faut ouvrir la balise <?php indiquant qu'on passe au php, et refermer avec ?> pour repasser au html

```
<?php
$affichage = 'accueil';
?>
[...] // Début de code html

<?php
if($affichage == 'jeu'){
    // On affiche ici le jeu
    ?>
    <table>[...]</table>
    <?php
}elseif($affichage == 'accueil'){
    // On affiche ici le formulaire pour le nom des joueurs
    ?>
    <form method="post">[...]</form>
    <?php
}else{
    // Une valeur par défaut quand la valeur n'est pas une des valeurs
prédéfinies
    ?>
    <h1>Erreur</h1>
    <?php
}
```

En réalité, dans ce genre de cas où on passe en revue la valeur d'une variable, on utilise plutôt la structure **switch**, le code suivant est totalement équivalent :

```
<?php
$affichage = 'accueil';

?>
[...] // Début de code html

<?php
switch($affichage){
    case 'jeu':
        // On affiche ici le jeu
        ?>
        <table>[...]</table>
```

```
        <?php
        break; // Important le break à la fin, sinon la suite sera exécutée
    case 'accueil':
        // On affiche ici le formulaire pour le nom des joueurs
        ?>
        <form method="post">[...]</form>
        <?php
        break; // Important le break à la fin, sinon la suite sera exécutée
    default:
        // Une valeur par défaut quand la valeur n'est pas une des valeurs
prédéfinies
        ?>
        <h1>Erreur</h1>
        <?php
}
```

L'intérêt de cette méthode est de séparer le code PHP de traitement du code html et php d'affichage : ceci permet d'avoir un code plus clair et lisible.

Dans le cadre de l'exemple on modifie à la main la variable bloc pour avoir les différents affichages, mais le but est que le traitement PHP définisse selon le stade de jeu ce qu'on affiche !

Le formulaire pour entrer les noms

Un formulaire se définit à l'aide de plusieurs balises HTML. Nous n'aurons pas besoin de toutes pour ce premier formulaire, mais uniquement de ces quelques premières :

- `<form method="post" action="morpion.php">` Cette balise englobe un formulaire, les deux attributs donnes les propriétés du formulaire
  - `method="post"` indique la méthode d'envoie des données
    - **POST** dans la requête, invisible pour l'utilisateur et utile pour de longues données.
    - **GET** dans l'URL (adresse de la page, après un ?), visible par l'utilisateur mais marche pour des données courtes
  - `action="morpion.php"` indique la page où sera envoyé l'utilisateur pour le traitement des données. On peut omettre ce paramètre si la page est la même. (**Ce qui sera notre cas pour le morpion**)
- `<input type="text" name="pseudo1" />` permet de faire un champ de texte court.
  - `type="text"` indique qu'il s'agit d'un texte court. Il y a plusieurs types selon les éléments du formulaire.
  - `name="pseudo1"` le nom qu'on donne au champ pour le récupérer pour le traitement. Il doit y en avoir un par champ.
  - `value="valeur par défaut"` valeur par défaut du champ.
  - `placeholder="Votre nom"` permet de mettre une valeur servant d'indicateur sur le champ. Comme une valeur par défaut mais sera effacée lorsque l'utilisateur cliquera sur le champ pour le remplir
- `<input type="submit" value="Commencer la partie" />` un bouton pour soumettre le formulaire
  - `type="submit"` indique qu'il s'agit d'un bouton de soumission du formulaire

    - `value="Commencer la partie"` sera le texte affiché sur le bouton… **Oui oui, sur l'attribue value**
- `Champ : <input type="text" id="champ" />` Permet de lier un texte de description à un champ. Quand on clique dessus le formulaire est sélectionner.
    - **Remarquez que le for="" et le id="" sont liés.**

Avec tout ça vous devriez être capable de faire un formulaire pour avoir les deux noms. Celui-ci devrait ressembler à ça :

```
<form method="post">
    <label>Joueur 1 : <input type="text" name="nom1"> </label>
    <label>Joueur 2 : <input type="text" name="nom2"> </label>
    <input type="submit" value="Commencer">
</form>
```

La zone de jeu

La zone de jeu sera simplement un tableau de 3 lignes et 3 colonnes. Pour faire un tableau on utilise quelques balises qui sont `<table>`, `<tr>` et `<td>` qui sont respectivement le tableau, la ligne de tableau et la case de tableau.

Un tableau contient des lignes qui contient des colonnes. Ainsi une grille vide avec que des [] en guise de case vide ressemblera à :

```
<table>
    <tr>
        <td><a href="?ligne=0&colonne=0">[]</a></td>
        <td><a href="?ligne=0&colonne=1">[]</a></td>
        <td><a href="?ligne=0&colonne=2">[]</a></td>
    </tr>
    <tr>
        <td><a href="?ligne=1&colonne=0">[]</a></td>
        <td><a href="?ligne=1&colonne=1">[]</a></td>
        <td><a href="?ligne=1&colonne=2">[]</a></td>
    </tr>
    <tr>
        <td><a href="?ligne=2&colonne=0">[]</a></td>
        <td><a href="?ligne=2&colonne=1">[]</a></td>
        <td><a href="?ligne=2&colonne=2">[]</a></td>
    </tr>
</table>
```

Etant donné qu'on va faire du traitement pour gérer l'affichage des cases, on peut utiliser des boucles en PHP pour gérer le tableau.

```
<table>
<?php
for($ligne = 0; $ligne < 3; $ligne++){
    echo '<tr>';
    for($colonne = 0; $colonne < 3; $colonne++){
```

```
        echo '<td><a href="?ligne='.$ligne.'&colonne='.$colonne.'">[]</td>';
    }
    echo '</tr>';
}
?>
</table>
```

Un lien commençant par ? point toujours vers la page courante. C'est un raccourci de **morpion.php?...** si la page s'appelle **morpion.php**.

Seconde partie : le traitement

On va maintenant s'occuper de ce code de début de page qui va faire le traitement. Il faut faire plusieurs choses :

- Déterminer sur quel “mode” on est : formulaire, jeu ou fin.
- En fonction du mode, écouter les bonnes entrées (données formulaire, case cliquée)
- Déterminer si un joueur a gagné

Retenir les données & savoir où on en est

Lorsqu'on va d'une page à une autre, toutes les données sont réinitialisées, pas de possibilité donc de mémoriser des choses facilement... Sauf avec les **sessions**. Les sessions nous permettent d'avoir des variables qui résistent au changement de page. Ici on va les utiliser pour stocker 4 variables :

- Le nom du joueur 1
- Le nom du joueur 2
- L'état de la grille (sous forme d'un tableau de tableau, une matrice de 3*3 quoi)
- Le tour actuel (qui du joueur 1 ou 2 doit jouer)

Pour pouvoir utiliser les sessions, il faut mettre au tout début du code :

```
<?php
session_start();
```

Il est important de le mettre au début du code, ou du moins avant tout code HTML et avant toute utilisation des sessions. Si il n'y a ne serait-ce qu'un espace de html avant l'appel de cette fonction, **ça ne fonctionnera pas** !

Les sessions l'utilisent simplement avec la variable `$_SESSION` qui est un array où on va venir assigner les valeurs qu'on veut aux clés qu'on veut.

On va utiliser le fait que ces variables existent ou non pour déterminer si on affiche le

formulaire ou le jeu. Une fois le formulaire validé on va créer ces variables.

```
<?php
session_start();

if(isset($_SESSION['nom1'], $_SESSION['nom2'], $_SESSION['grid'],
$_SESSION['tour'])){
   // Le code de traitement pour le jeu
   $affichage = 'jeu';
}else{
  // Le code de traitement pour le formulaire
  $affichage = 'accueil';
}

[...] // Le code d'affichage
```

Récupérer les données du formulaire

Les données du formulaire sont stocké dans une des deux variables `$_GET` et `$_POST` selon la méthode choisie. Ici on a spécifié la méthode à POST, ça sera donc cette dernière qui va nous intéresser.

Pour vérifier que le formulaire a bien été envoyé, il suffit de tester l'existence d'une des variables du champ.

```
if(isset($_POST['nom1']) && isset($_POST['nom2'])){
    // Traiter le formulaire
}else{
    // Afficher le formulaire
}
```

Le nom de la valeur qu'on récupère dans `$_POST` correspond au nom donné dans l'attribut **name** de la balise **input**.

Une fois les données crées, il suffit d'initialiser les 4 variables de jeu pour lancer une partie.

Pour un peu plus de fun, on choisit aléatoirement quel joueur commence avec la fonction `mt_rand(1,2)` qui renvoie un entier au hasard entre 2 bornes (ici 1 et 2...)

A la fin du traitement, il faut relancer le script, pour que la condition vérifiant si les variables existe soit exécutée et ainsi afficher la grille. Une autre solution est de faire deux **if** indépendants pour ces deux vérifications. Pour faire une redirection : `header('location: page.php');` et ne pas oublier n'arrêter le script après la redirection avec l'instruction `exit;`

Le code de traitement devrait donc jusque là ressembler à ça :

```
<?php
session_start();
```

```
if(isset($_SESSION['nom1'], $_SESSION['nom2'], $_SESSION['grid'],
$_SESSION['tour'])){
    // Le code de traitement pour le jeu
    $affichage = 'jeu';
}else{
    // Le code de traitement pour le formulaire
    if(isset($_POST['nom1'], $_POST['nom2'])){
        $_SESSION['nom1'] = $_POST['nom1'];
        $_SESSION['nom2'] = $_POST['nom2'];
        $_SESSION['tour'] = mt_rand(1, 2);
        $_SESSION['grid'] = [
            [0, 0, 0],
            [0, 0, 0],
            [0, 0, 0]
        ];
        header('location: ./morpion.php'); // On redirige vers la même page
pour relancer le code.
            exit; // On arrête le script pour que la suite ne soit pas
exécutée
    }else{
        $affichage = 'accueil';
    }
}
```

Si votre fichier ne s'appelle pas **morpion.php**, changez la ligne avec le `header('location: ./morpion.php');`

Afficher nos belles données

Il serait dommage de ne pas afficher ces données récupérées non ?

Pour afficher le nom du joueur actuel, il suffit d'afficher `$_SESSION['nom1']` ou `$_SESSION['nom2']` en fonction du contenu de `$_SESSION['tour']` (qui vaut 1 ou 2).

simplement, on a :

```
echo '<h3>C\'est au tour de
'.htmlspecialchars($_SESSION['nom'.$_SESSION['tour']]).'</h3>';
```

On applique toujours la fonction `htmlspecialchars($var)` sur une variable venant de l'utilisateur qu'on affiche. Cela lui empêche d'injecter son propre code HTML : **C'est une des fonctions de base pour la sécurité de vos sites***

Et pour jouer ?

Il est temps de pouvoir placer ses croix et ses ronds !

Pour rappel, lorsque qu'on veut jouer on se rend sur le lien *morpion.php?**ligne=0**&**colonne=2***. Les variables dans l'URL se récupèrent avec la variable `$_GET`, ici `$_GET['ligne']` et `$_GET['colonne']`. Il faudra vérifier si aux coordonnées données la valeur est bien à 0 (aucun joueur n'a déjà joué). Ensuite il faut appliquer la valeur de `$_SESSION['tour']` (qui contient le numéro du joueur dont c'est le tour) et changer cette valeur.

Après avoir fait ça le code de traitement devrait ressembler à ça :

```php
<?php
if(isset($_SESSION['nom1'], $_SESSION['nom2'], $_SESSION['grid'], $_SESSION['tour'])){
    $affichage = 'jeu';

    if(isset($_GET['ligne'], $_GET['colonne'])){
        $l = $_GET['ligne'];
        $c = $_GET['colonne'];

        if($_SESSION['grid'][$l][$c] != 0) exit('Petit coquin...'); // Un brin de vérification

        $_SESSION['grid'][$l][$c] = $_SESSION['tour'];

        $_SESSION['tour'] = ($_SESSION['tour'] == 1) ? 2 : 1; // Forme ternaire : $variable = condition ? valeur si vraie : valeur si fausse;
        header('location: ./morpion.php');
            exit;
    }

}else{
    if(isset($_POST['nom1'], $_POST['nom2'])){
        $_SESSION['nom1'] = $_POST['nom1'];
        $_SESSION['nom2'] = $_POST['nom2'];
        $_SESSION['tour'] = mt_rand(1, 2);
        $_SESSION['grid'] = [
            [0, 0, 0],
            [0, 0, 0],
            [0, 0, 0]
        ];
        header('location: ./morpion.php');
            exit;
    }else{
        $affichage = 'accueil';
    }
}
```

Si votre fichier ne s'appelle pas **morpion.php**, changez la ligne avec le `header('location: ./morpion.php');`

Maintenant pour que ça marche il faut modifier le code d'affichage du tableau pour afficher une X ou un O en fonction de la valeur stockée pour la grille.

Les boucles affichant le tableau fait parcourir les valeurs 0, 1 et 2 aux variables `$ligne` et `$colonne`, il suffit donc de faire une simple vérification.

```
for($ligne = 0; $ligne < 3; $ligne++){
    echo '<tr>';
    for($colonne = 0; $colonne < 3; $colonne++){
        if($_SESSION['grid'][$ligne][$colonne] == 0)
            echo '<td><a
href="?ligne='.$ligne.'&colonne='.$colonne.'">[]</a></td>';
        elseif($_SESSION['grid'][$ligne][$colonne] == 1)
            echo '<td>X</td>';
        else
            echo '<td>O</td>';
    }
    echo '</tr>';
}
```

On enlève les liens quand une case est jouée : on ne doit pas rejouer dessus !

Vérifier si il y a un gagnant

Il faut maintenant écrire un code vérifiant si un des joueurs a gagné. Pour cela il faut faire des conditions sur les valeurs de `$_SESSION['grid']`

Il y a 8 cas de victoire :

- 3 en ligne
- 3 en colonne
- 2 en diagonale

Et un cas de match nul : toutes les cases sont remplies.

Pour vérifier qu'il y a un gagnant sur une ligne il faut vérifier que les 3 cases aient la même valeur et qu'elles soient différentes de zéro (sinon y'a victoire dès le début...). Par exemple :

```
if($_SESSION['grid'][0][0] != 0 && $_SESSION['grid'][0][0] ==
$_SESSION['grid'][0][1] && $_SESSION['grid'][0][1] ==
$_SESSION['grid'][0][2]){
    // Victoire sur la première ligne
}
```

Bien évidemment on ne va pas écrire 8 lignes comme ça, on peut utiliser des boucles pour réduire le tout à 4 conditions (sans compter la vérification du match nul...)

Personnellement, j'a écrit une fonction `is_there_a_winner()` qui peut renvoyer 4 valeurs :

- 1 ou 2 si un des joueurs a gagné
- -1 s'il y a un match nul
- 0 si la partie continue

```
function is_there_a_winner(){
    $grid = $_SESSION['grid'];
    // Vérifications sur les lignes et colonnes :
    for($i = 0;$i<3;$i++){
        if($grid[$i][0] == $grid[$i][1] && $grid[$i][1] == $grid[$i][2] &&
$grid[$i][0] != 0) return $grid[$i][0];
        if($grid[0][$i] == $grid[1][$i] && $grid[1][$i] == $grid[2][$i] &&
$grid[0][$i] != 0) return $grid[0][$i];
    }

    // vérifications sur les 2 diagonales :
    if($grid[0][0] == $grid[1][1] && $grid[1][1] == $grid[2][2] &&
$grid[1][1] != 0) return $grid[1][1];
    if($grid[0][2] == $grid[1][1] && $grid[1][1] == $grid[2][0] &&
$grid[1][1] != 0) return $grid[1][1];

    // Vérification de match nul : est-ce que la grille est remplie N
    $nb0 = 0; // On compte les zéros restants
    for($i = 0;$i<3;$i++){
        for($j = 0;$j<3;$j++){
            if($grid[$i][$j]==0)
                $nb0++;
        }
    }

    if(!$nb0) return -1;

    return 0;
}
```

Une fois cette fonction écrite, il suffit de l'appeler en cours de jeu pour vérifier si il faut continuer ou arrêter le jeu. Arrêter le jeu pourra se résumer à afficher une vue spécifique.

```
$winner = is_there_a_winner();
if($winner != 0){ // Il se passe quelque chose
    $affichage = 'resultats';
}else{
    // Vérifier si il y a un coup joué...
}
```

Le fait de stocker le résultat du calcul dans `$winner` permet d'éviter l'appel de

`is_there_a_winner()` plusieurs fois, et donc d'éviter de faire plusieurs fois inutilement le même calcul. Ici le gain est négligeable, mais c'est une habitude à prendre avec les fonctions faisant du traitement.

Il faudra ensuite modifier la vue en conséquence.

```
case 'resultats':
    if($winner > 0){
        echo '<h1>Le joueur '.htmlspecialchars($_SESSION['nom'.$winner]).' a
gagné ! </h1>';
    }else{
        echo '<h1>Match nul...</h1>';
    }
    break;
```

Rappel : On applique toujours la fonction `htmlspecialchars($var)` sur une variable venant de l'utilisateur qu'on affiche. Cela lui empêche d'injecter son propre code HTML : **C'est une des fonctions de base pour la sécurité de vos sites***

Rejouer

Dans l'état, le jeu est bloqué à une seule partie. Pour recommencer, il suffirait de détruire 4 variables pour retourner à l'écran de sélection des noms.

La fonction `unset($variable)` permet de supprimer une variable. Il suffit donc de l'appliquer à nos 4 variables de session à la fin d'une partie pour réinitialiser le jeu.

```
unset($_SESSION['nom1']);
unset($_SESSION['nom2']);
unset($_SESSION['grid']);
unset($_SESSION['tour']);
```

Ou encore plus simple, la fonction `session_destroy()` supprimera toute trace des variables de session.

Et voila, c'est fini

Correction

Normalement si vous avez bien suivi et compris ce tuto, votre version devrait marcher. Mais si vous voulez une correction, demandez moi. (rgrondin)

Idées d'améliorations

On a maintenant une première version basique du morpion. Bien entendu on peut facilement l'améliorer, voici quelques idées :

- Rajouter à coté du nom du joueur si il joue avec les X ou les O.
- Afficher les deux noms des joueurs avec les X et les O
- Après une fin de partie, retenir les noms des joueurs et les mettre par défaut dans le formulaire (Il ne faut donc pas utiliser la fonction `session_destroy()` et détruire sélectivement les variables.)
- Mettre en place un système de scores (Qui durera que dans le temps de la session du coup)
- Faire une IA contre laquelle jouer (On passe à un autre niveau là)

From:
https://wiki.centrale-med.fr/ginfo/ - **Wiki GInfo**

Permanent link:
https://wiki.centrale-med.fr/ginfo/formations:devweb_3_exos_31

Last update: **20/10/2020 14:27**