



## 12) Traitement des formulaires

Nous avons pour l'instant toujours utilisés le terminal pour interagir sur un script/programme que l'on créait en PHP. On utilisait la fonction `readline()`, qui permettait de lire au clavier, puis on traitait l'information.

Ici on va utiliser les formulaires pour faire dans un premier temps exactement ce que l'on faisait dans le terminal. Ensuite plus tard on utilisera bien entendu les formulaires pour interagir avec notre Base de Données (exemple inscription sur un site, login etc.).

Pour commencer on va reprendre un des exercices qu'on avait fait durant la formation avec la fonction `rand()`, qui permettait de deviner un chiffre compris entre 0 et 10. On va bien entendu l'améliorer un peu.

```
$nombre=(int)readline("Entrez un nombre entre 0 et 10 pour gagner un lot : ");
$numeroGagnant = rand(0,10);
while($nombre != $numeroGagnant){
    echo "Mauvais numéro, vous n'avez pas gagné !". "\n";
    $nombre=(int)readline("Retentez votre chance, entrez un nombre à
nouveau : ");
}
echo "Bravo !!! \nVous avez enfin trouvé le numéro gagnant ! \nC'était le
numéro $numeroGagnant";
```

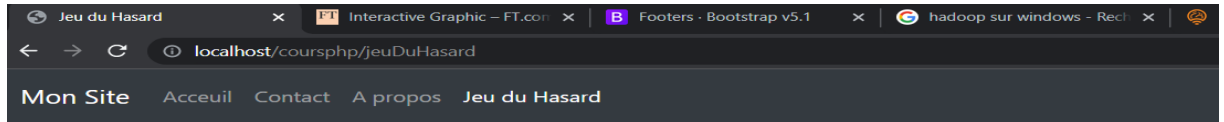
On va créer un fichier qu'on va appeler **jeuDHasard.php**, dans ce fichier on va un peu s'inspirer de ce qu'on avait fait dans le chapitre des boucles il y a quelques mois.

Tout d'abord on va créer une variable **\$numeroGagnant** qui reçoit un nombre entier **aléatoire** compris entre **0 et 10**.

Pour l'instant en php on utilisait la fonction `readline()` pour pouvoir interagir avec le programme. Ici avec HTML on va le faire via des formulaires. Vous avez déjà vu comment en créer dans votre cours de HTML, donc je considère que c'est acquis.

## (Exos)

- 1) Créez un onglet jeu du hasard. Faites comme on a fait pour les autres onglets donc quand on clique sur jeu du hasard ça le met en surbrillance et on voit qu'on est sur la page Jeu du hasard(title).



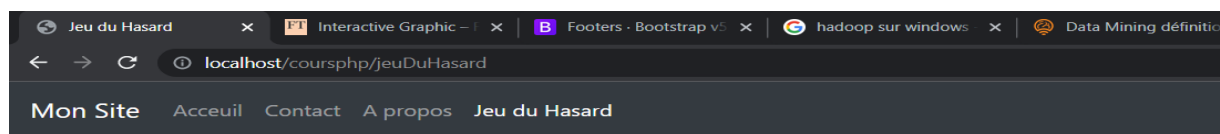
[Accueil](#) [Contact](#) [A propos](#)

© 2022 Cfitech, Inc

On va maintenant créer notre formulaire avec le mot clé « form ».

```
<form action="/coursphp/jeuDuHasard.php" method = "GET">
  <input type="number" name="nombre" placeholder="Entre 0 et 10">
  <button type="submit">Deviner</button>
</form>
```

Si maintenant je vais sur ma page web, on aura ce résultat :

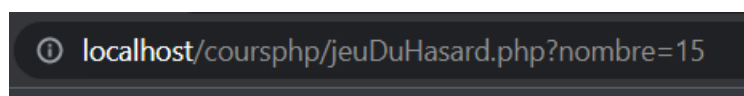


Entre 0 et 100

[Accueil](#) [Contact](#) [A propos](#)

© 2022 Cfitech, Inc

Vous remarquerez que si vous entrez un nombre, il relancera la page Jeu du Hasard mais il y aura une différence dans le lien. L'URL aura changé. On verra notre variable nombre qu'on a créé en HTML avec le nombre qu'on aura introduit.



Ici j'avais introduit 15, du coup notre variable nombre reçoit 15.

On aura donc besoin de récupérer ce champ pour pouvoir le comparer avec notre numéro gagnant.

On va le récupérer grâce à une variable globale qui pour la méthode GET s'appelle `$_GET["nomDeLaVariableARecuperer"]`. C'est en php, il ne faut donc pas oublier de mettre les balises php.

Je vais vous donner directement le code.

```
<?php if ($_GET['nombre'] > $numeroGagnant): ?>
    Votre nombres est trop grand !
<?php elseif ($_GET['nombre'] < $numeroGagnant): ?>
    Votre nombres est trop petit !
<?php else: ?>
    Vous avez trouver le numéro gagnant : <?php echo $numeroGagnant; ?>
<?php endif;?>
```

Essayez ce code, que constatez-vous ?

### (Exos)

- 2) Essayez d'améliorer ce code pour qu'on ait plus d'erreur, quand on lance la page jeu de hasard. Comment le faire ?
- 3) Améliorez encore le code pour faire en sorte que quand on introduit des valeurs non comprises entre 0 et 10 il renvoi un message adapté.

Certains le savent déjà mais ici nous avons utiliser la **méthode GET** pour récupérer le nombre dans nos formulaires, il était ensuite montrer dans notre URL. Il existe une autre méthode pour récupérer les données d'un formulaire et c'est avec la **méthode POST**.

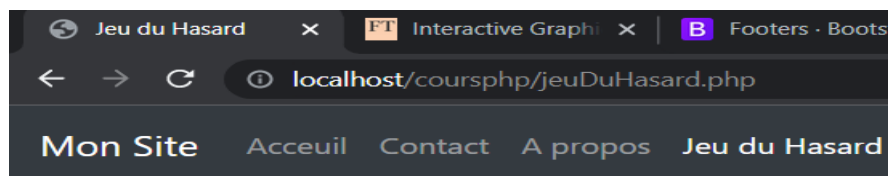
Il fonctionne de la même manière que le GET.

On va aller modifier notre page du jeuDuHasard :

```
<form action="/coursphp/jeuDuHasard.php" method = "POST">
```

Et il faut aussi aller modifier la variable locale ce ne sera plus `$_GET[]` mais `$_POST[]`.

Maintenant quand vous introduirez un nombre, on ne verra rien dans l'URL, il est donc un peu plus safe que le GET parce qu'avec le GET on aurait pu tout simplement modifier le lien pour changer la valeur.



Vous avez bien trouvé le bon nombres 3

Entre 0 et 10

Vous aurez compris que la plupart des sites, pour ce qui est des formulaires avec des données sensibles, utiliserons un POST.

**Pour résumer**, retenez que dans un formulaire on peut utiliser la méthode GET ou la méthode POST pour récupérer les données que l'utilisateur introduira. En PHP on utilisera `$_GET[]` ou `$_POST[]`, ce sont des variables globales. Ce sera pareil pour la manière de se connecter avec un Login et un password, c'est aussi via des formulaires. Vous aurez compris que pour des données sensibles comme les passwords, on utilisera un POST pour qu'on ne puisse pas voir le mot de passe dans l'URL.

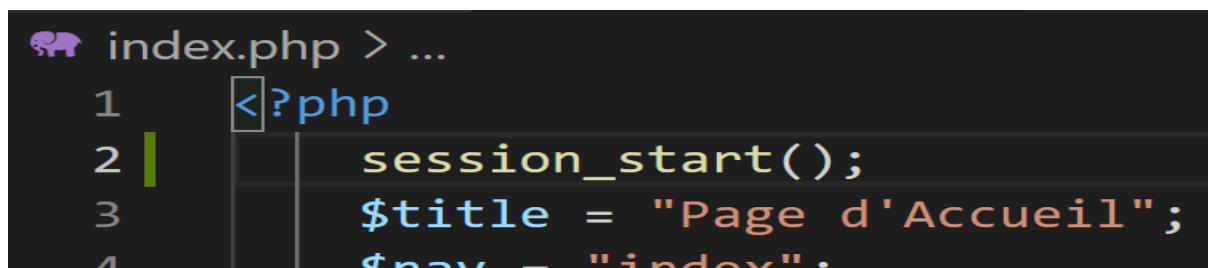
## 13) Les Sessions

Nous allons découvrir ensemble le principe des sessions en PHP. Donc les sessions vous pouvez imaginer ça comme une sorte de cookies qui serait top secret.

Pour utiliser une session en PHP, il suffit d'utiliser une fonction qui s'appelle « **session\_start()** ». Vous la retrouvez bien entendu dans la documentation officielle sur php.net. Ça permet de **démarrer ou reprendre une session**.

On pourra l'utiliser pour définir des rôles, comme administrateur, utilisateurs ou abonnés et en fonction du rôle, avoir des onglets en plus ou en moins etc.

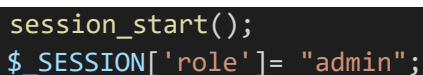
On va l'utiliser dans notre « **index.php** », on le met au tout début :



```
index.php > ...
1  <?php
2  session_start();
3  $title = "Page d'Accueil";
4  $nav = "index";
```

Une fois que vous avez utilisé cette fonction, vous allez pouvoir modifier une super variable globale qui s'appelle « **\$\_SESSION[]** ». Une variable globale est une variable qui est accessible partout/globalement dans le script.

Cette variable globale vous permettra de lire mais aussi de modifier les informations qui seront sauvegardé en session. Par exemple si je veux définir le rôle d'un utilisateur en tant qu'administrateur, ce n'est pas quelques choses que je peux modifier dans les cookies sinon n'importe qui pourrait aller les modifier. On va plutôt utiliser le **\$\_SESSION** qui fonctionne comme un tableau donc je peux créer une clé « rôle » et sauvegarder le rôle administrateur :



```
session_start();
$_SESSION['role'] = "admin";
```

C'est tout ce que j'ai à faire à ce niveau-là. Si maintenant dans une autre page, j'ai besoin de récupérer les informations de la session, je dois refaire **session\_start()**. Ensuite là où je le souhaite on va pouvoir récupérer les informations.

Je vais rajouter dans le fichier « **contact.php** » la fonction **session\_start()** qui je le rappelle démarre ou reprend une session.

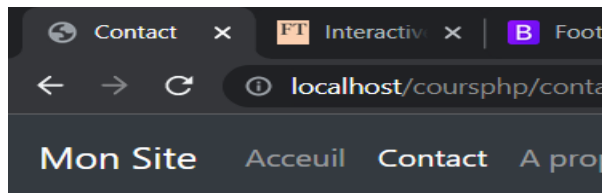
```

contact.php > p
1  <?php
2      session_start();
3      $title = "Contact";
4      $nav = "contact";
5      require "header.php";
6  ?>
7  <h1>Debug</h1>
8  <p><?php var_dump($_SESSION)?></p>

```

Ici on va utiliser la fonction **var\_dump()** en php, je vous avais dit qu'on l'utiliserait surtout pour déboguer et voir que contiennent les variables.

Ça donnera ce résultat après être passé sur la Page d'Accueil.



## Debug

```

C:\wamp64\www\coursphp\contact.php:8:
array (size=1)
    'role' => string 'admin' (length=5)

```

## Nous contacter

On voit bien ici que nous avons un tableau indexé de taille 1, avec comme clé 'role' et comme valeur une chaîne de caractères « admin » de taille 5.

Il a donc bien gardé la session, il faut savoir que les sessions ne vivent que pendant une navigation de l'utilisateur, c'est-à-dire que si l'utilisateur ferme son navigateur, la session sera automatiquement perdue. Exemple Banque.

On peut sauvegarder tout type d'information (chaîne de caractères, entiers, réels, objets, tableaux, etc.).

On va voir que c'est possible de récupérer le même style de donnée qu'on avait créé dans le chapitre sur les tableaux indexés.

Vous vous rappelez de ce tableau indexé :

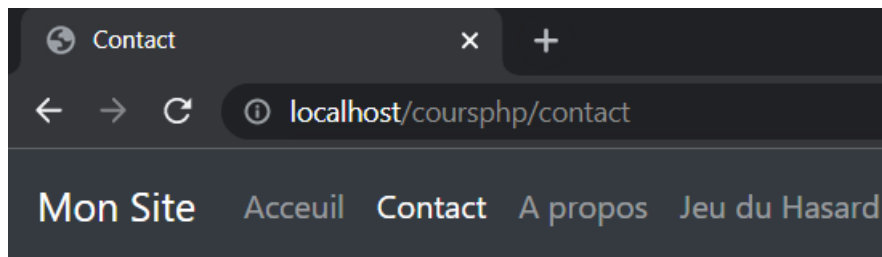
```
$users = [
    [
        "Prenom" => "Sarah",
        "Sexe"    => "F",
        "Age"     => 35
    ],

```

On peut le sauvegarder par exemple dans une session, je vais un peu le modifier.

```
session_start();
$_SESSION['user'] = [
    'firstname' => "Julien",
    'lastname'  => "Dunia",
    'login'     => "deoking",
    'password'  => 1111
];
```

Et quand je l'exécute via le var\_dump :



## Debug

C:\wamp64\www\coursphp\contact.php:8:

```
array (size=1)
  'user' =>
    array (size=4)
      'firstname' => string 'Julien' (length=6)
      'lastname'  => string 'Dunia' (length=5)
      'login'     => string 'deoking' (length=7)
      'password'  => int 1111
```

On voit bien que notre session a su garder notre utilisateur avec son prénom, son nom, son login et son mot de passe.

Dans la suite nous allons essayer de faire ensemble un système d'authentification. En fonction de si on est un admin ou un utilisateur normal on voit différemment. On ne fait pas encore la connexion à la base de données mais on y arrive.