

# Première partie

## Architecture HTML CSS

Vous allez développer une application web simulant un jeu de roulette. L'application utilisera HTML, CSS, PHP, une base de données phpMyAdmin et éventuellement d'autres bibliothèques de votre choix (par exemple Bootstrap pour le CSS).

Les notions abordées par le TP Roulette sont les suivantes:

- développement web côté client (HTML)
- développement web côté serveur (PHP)
- envoi d'informations d'une page à l'autre (formulaires GET et POST)
- accès et gestion de base de données SQL (phpMyAdmin)
- gestion de sessions PHP
- architecture MVC

Le TP est divisé en plusieurs parties. Dans cette première partie, vous vous concentrerez sur l'architecture HTML des trois modules de l'application: le module de connexion, celui d'inscription et celui du jeu.

### I. La roulette

La roulette est un jeu de hasard. Elle comporte 36 cases numérotées de 1 à 36, une bille est lancée dedans et s'arrête à un moment dans une case. Le but est de prévoir où la bille va s'arrêter en misant sur le résultat. Il existe plusieurs façon de miser offrant plus ou moins de chance de gagner et les gains s'accordent avec le pourcentage de chance de gain.

Avant chaque partie, le joueur mise de l'argent selon un choix de jeu. Dans notre cas, les possibilités de jeux sont limitées aux suivantes:

- Le joueur peut choisir de miser sur la valeur exacte du résultat
- Le joueur peut choisir de miser sur la parité du résultat

En fonction du choix fait par le joueur, deux gains sont possibles:

- 35 fois sa mise si il trouve la valeur exacte
- 2 fois sa mise si il trouve la bonne parité

### II. Les modules de l'application

L'application est divisée en trois modules, vous aurez donc 3 fichiers:

- *connexion.php* : le module de connexion
- *inscription.php* : le module d'inscription
- *roulette.php* : le module de jeu

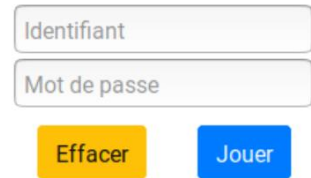
Vous pouvez renommer le fichier *connexion.php* en *index.php*, c'est sur ce fichier que l'utilisateur devra arriver en premier.

### A. Connexion

Ce module est composé d'un formulaire comportant:

- un champ identifiant
- un champ mot de passe
- un bouton de validation
- un éventuel bouton permettant d'effacer le contenu des champs

Connectez-vous pour  
jouer à la roulette



Nous ajouterons plus tard du code PHP pour gérer la connexion.

Pour l'instant, vous pouvez faire pointer le formulaire (attribut *action*) sur le fichier *roulette.php*. Réfléchissez sur l'utilisation de la méthode (GET ou POST ?). Faites des tests et observez l'URL.

### B. Inscription

Au niveau de l'architecture HTML, le module d'inscription ressemble grandement au module de connexion: il contient deux champs de texte et un bouton de validation. De même, le formulaire pointe sur ce même fichier *inscription.php*

### C. Roulette

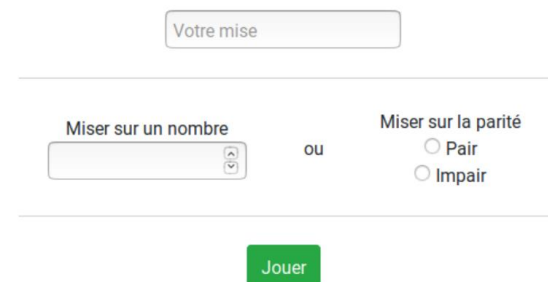
La page *roulette.php* comporte un formulaire disposant:

- d'un champ où l'utilisateur entre sa mise
- d'un champ où l'utilisateur choisit un nombre de 1 à 36
- deux boutons radio: pair ou impair, si l'utilisateur mise sur la parité
- un bouton d'envoi du formulaire

Nous verrons dans le prochain TP comment nous allons implémenter le jeu.

Le jeu de roulette  
Le jeu de roulette

bthouverez



### D. Déconnexion

L'utilisateur doit pouvoir se déconnecter lorsqu'il joue à la roulette. Il s'agit alors d'ajouter un lien vers la page de connexion.

Si vous avez fini, essayez de rendre votre site joli avec du CSS, regardez les éventuelles bibliothèques CSS (Bootstrap, Materialize, Knacss...).

Dans la prochaine partie, nous verrons comment récupérer les données issues des formulaires et les sauvegarder.

## Deuxième partie

### Transmissions de données, sessions et jeu de la roulette

Dans cette partie, vous allez récupérer les données issues de vos formulaires et les sauvegarder dans une session.

#### - Récupération des données des formulaires

##### - Connexion

Le formulaire de connexion doit dorénavant pointer sur le même fichier: *connexion.php* (ou *index.php*). Vous traiterez les données issues des formulaires (tableaux *\$\_GET*, *\$\_POST* et méthode *isset*). Dans un premier temps, vous comparerez les identifiants et mots de passes avec des valeurs codées en dur dans l'application. Si les données du formulaire sont correctes, vous redirez l'utilisateur vers le jeu de roulette avec la méthode *header* après avoir retenu les informations de l'utilisateur dans une session (tableau *\$\_SESSION*): vous retiendrez le nom d'utilisateur et son argent. Dans le prochain TP vous utiliserez une base de données stockant les données utilisateur.

##### - Inscription

Sur le même principe, le formulaire doit pointer sur le fichier *inscription.php*. Pour l'instant, remplissez la session et redirez l'utilisateur vers le jeu de roulette. Plus tard, il s'agira en plus d'insérer les données du nouvel utilisateur dans la base de données.

##### - Déconnexion

La déconnexion nécessite simplement de vider la session pour supprimer les données de l'utilisateur en cours. Un simple lien ne suffit plus, vous devez transmettre l'information que vous cherchez à vous déconnecter, personnellement, j'aime bien la méthode suivante:

```
<!-- le lien de déconnexion ... -->
<a href="index.php?deco">Déconnexion</a>
<!-- qui fait une simple requête GET vers index.php -->

<?php # dans le fichier index.php
if(isset($_GET['deco'])) {
    # mon utilisateur se déconnecte
    # je vide la session
}
?>
```

Pas besoin de valeur à l'attribut *deco*, on vérifie juste s'il existe. De cette manière on peut directement se déconnecter en ajoutant *?deco* dans l'URL.

#### - Sécurisation des pages

Vous utiliserez la session pour protéger vos pages:

- si l'utilisateur tente d'accéder au jeu de roulette sans avoir été identifié au préalable, vous le redirigez vers la connexion
- si un utilisateur déjà connecté tente d'accéder à la connexion, vous le redirigez vers le jeu
- si un utilisateur déjà connecté tente de s'inscrire à nouveau, vous récupérez les données du formulaire et changez la session pour le nouvel utilisateur inscrit

## - Le jeu de la roulette

Pour jouer, l'utilisateur doit:

III. Miser une somme d'argent, inférieure à l'argent qu'il détient

IV. Choisir de miser sur un nombre de 1 à 36

V. Ou choisir de miser sur la parité: on utilisera des radio buttons pour gérer cela

Le ou est exclusif, le joueur ne peut miser qu'avec une possibilité lors d'une partie. On partira du principe que, du moment que le joueur inscrit un numéro de case, il mise sur cette case, même si un bouton radio pour la parité a été coché.

Le formulaire pointe sur le même fichier. Quand l'utilisateur le validera, vous veillerez à vérifier dans un premier temps si la partie peut être jouée:

- l'utilisateur a rempli sa mise et les champs d'une possibilité de jeu
- la mise est inférieure à la somme d'argent détenue par le joueur

Si la partie peut être jouée, le déroulement est le suivant:

- le joueur perd sa mise
- la roulette est lancée, la bille s'arrête sur une case (pour simuler ceci, on effectuera un tirage aléatoire d'un nombre entre 1 et 36 grâce à la fonction *rand* de PHP, Cf la doc).
- selon le jeu choisi par l'utilisateur s'il gagne en ayant misé sur un nombre exact, il gagne 35 fois sa mise, si il gagne avec la parité, il gagne 2 fois sa mise, sinon il ne gagne rien.

Sinon, vous informerez l'utilisateur des erreurs qu'il a commis.

Dans tous les cas, faites apparaître des messages d'information du type :

- Vous avez misé sur le 7, hélas c'est le 34 qui est tombé, vous avez perdu
- Vous pensiez que le résultat serait pair et le 34 a été tiré. Bravo, vous gagnez 54€

Ils vous aideront fortement pour déboguer vos scripts.

Dans le prochain TP, vous prendrez en main phpMyAdmin pour sauvegarder vos données dans une base de données.

## Quatrième partie

### Programmation orientée objet

Le but de cette partie est d'appréhender la programmation orientée objet en PHP.

#### - Un objet pour gérer notre base de données

Créez une classe qui regroupera les fonctions d'accès à la base de données. Elle contiendra au minimum comme attributs:

- VI. un attribut *bdd* qui sera l'objet PDO
- VII. une chaîne de caractère pour stocker l'hôte de la base
- VIII. une chaîne de caractère pour stocker le nom de la base
- IX. une chaîne de caractère pour stocker le nom d'utilisateur
- X. une chaîne de caractère pour stocker le mot de passe

En plus du constructeur et des habituels getters et setters, elle contiendra les méthodes suivantes:

- une méthode d'initialisation qui initialisera l'objet PDO
- une méthode de connexion de l'utilisateur
  - vérification des données
  - remplissage de la session
  - retourne un éventuel message d'erreur
- une méthode d'ajout d'un utilisateur dans la base de données
- une méthode de mise à jour d'un joueur dans la base de données
- une méthode d'ajout de partie dans la base de données

#### - Un objet pour gérer une partie

Il peut aussi être intéressant de créer un objet qui va gérer une partie. Cela permettra de regrouper le code à l'intérieur d'une classe.

Plus tard, nous regrouperons ces classes dans une arborescence précise pour un premier pas vers une architecture MVC. Elles feront notamment parti du Modèle qui contient tout ce qui est relatif aux traitements et aux calculs du site web ainsi que tout ce qui concerne l'accès aux données.