Anthony Laplane

Projet - PHP

studi

Déroulé

- **7 lives d'une heure** sur la mise en place d'un **projet en PHP**.
  - **On peut déjà s'inscrire aux prochains**.
- Ce live s'adresse aux débutants en php (pas de poo)
- Prérequis :
  - Avoir les connaissances en HTML/CSS, les **bases** de PHP et MySQL (PDO)
- Nous n'allons pas aborder les notions de POO (objet)
- Questions/Réponse à la moitié (25min) et fin du live (55min).
- Sources du projet sur github : https://github.com/arirangz/studi_allgames

Programme

- Présentation du projet
- Conception de la base de données
- Mise en place des templates avec **Tailwind**
- Inscription/connexion
- Lister les jeux
- Consulter un jeu et pouvoir l’ajouter à sa liste de souhaits (wishlist)
- Pouvoir laisser une critique du jeu
- Mise en place de la recherche

Environnement technique

- PHP >= 8.1 et MYSQL >= 5.7 (WAMP)
- Tailwind
- VSCode

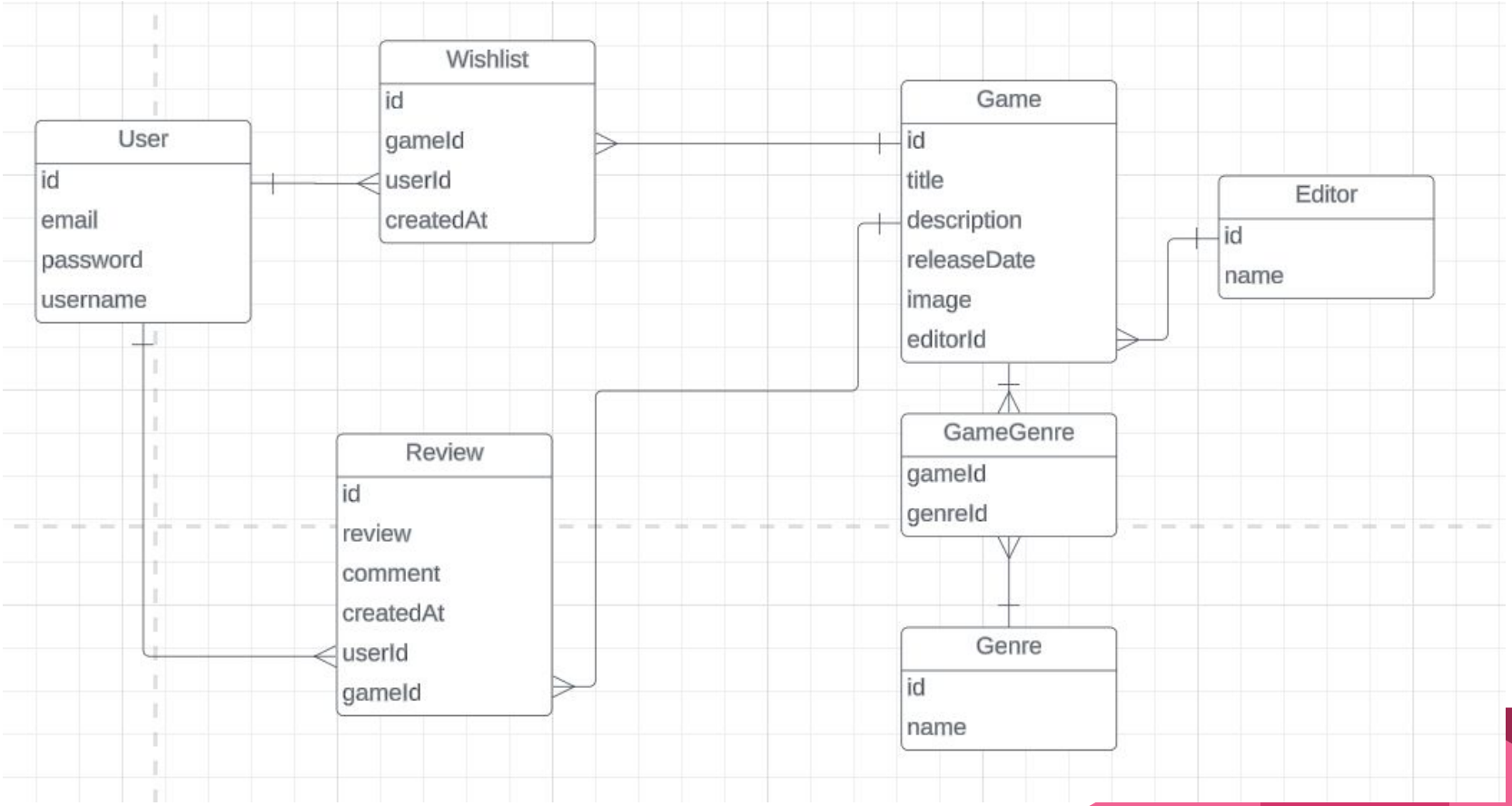
Besoin

- La société AllGames veut proposer un site simple permettant aux joueurs de créer leur liste de souhaits de jeux vidéo
- Elle souhaite que les utilisateurs puissent s'inscrire (email, mot de passe, pseudo)
- Les utilisateurs pourront rechercher un jeu, consulter la fiche du jeu et l'ajouter à leur liste de souhait. Ils pourront également laisser une critique du jeu (positif/négatif -> bool, détail de la critique)
- On affichera sur la page d'accueil les genres de jeux et les derniers jeux
- Une page devra lister les jeux présents dans la liste de souhaits
- Les jeux sont associés à un éditeur et peuvent être associés à plusieurs genres

Charte graphique - Tailwind

- Police : system-ui
- Couleur : blue-500 gray-900
- Logo :

Diagramme Entité-Relation



LES FONDAMENTAUX

Introduction

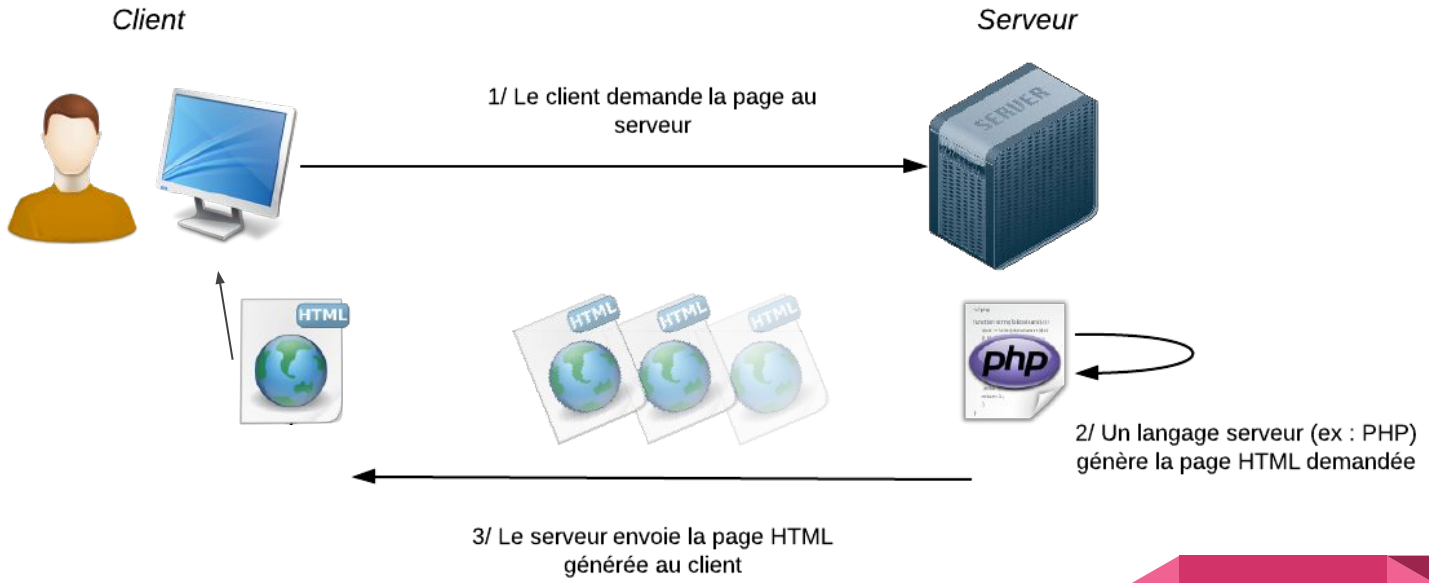
PHP est un langage de programmation tout comme javascript à la différence que javascript est un langage client (lorsqu'il est utilisé sur une page web) alors que php est un langage serveur.

Langage exécuté côté client : Lorsque le code est exécuté, cela se passe sur l'ordinateur de l'utilisateur (le client)

Langage exécuté côté serveur : Lorsque le code est exécuté, cela se passe sur le serveur

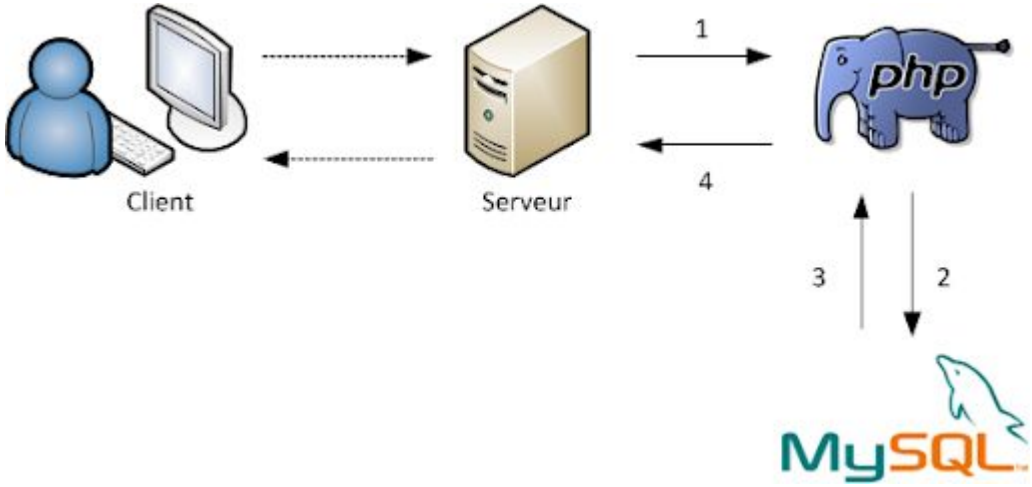
Introduction

PHP va donc être exécuter sur le serveur, avant même que la page soit affichée à l'écran.



Introduction

Pour fonctionner, php a besoin d'un serveur web (ex: apache) et d'une base de données (dans la plupart des cas).



Introduction

Le langage php permet de créer des sites dynamiques avec des données issues d'une base de données (ex: mysql).

La plupart des CMS utilisent le langage PHP (Wordpress, Drupal, Prestashop, Joomla etc.).

Plus de 75% des sites utilisent PHP.

Quelques alternatives à PHP :

Fichier php

Un fichier php se termine par l'extension php (ex: page.php).
Le code php que l'on va écrire dans un fichier php va être entre une balise d'ouverture **<?php** et de fermeture **?>**

```php
<?php

//je peux écrire entre les balises le code php

?>
```

Fichier php

Un fichier php va souvent mélanger du php et du html. Le html doit être écrit en dehors des balises php

```
<h1>Titre en html</h1>

<?php

//je peux écrire entre les balises le code php

?>

<h2>Titre en html</h2>
```

Echo

L’instruction echo va nous permettre d’afficher un message sur la page

```php
<?php
echo 'Bonjour';
// ou
echo('Bonjour');
?>
```

Commentaires

Comme en javascript, on peut ajouter des commentaires dans notre code pour le rendre plus lisible.

```php
<?php
// Commentaire sur une ligne


/*
 Un commentaire qui pourra
 contenir plusieurs ligne
*/
?>
```

Variables

Une variable est une zone mémoire qui pourra contenir une valeur. C’est une sorte de boîte dans laquelle on va pouvoir stocker une information.

```php
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On affiche le contenu de la variable age sur la page avec echo
echo $age;

?>
```

Constantes

En PHP, une constante est une valeur qui ne peut pas être modifiée pendant l'exécution du script. Une constante est définie grâce à la fonction "define()", qui prend deux arguments : le nom de la constante et sa valeur.

```php
<?php

   // Une constante n'est défini qu'une seule fois
   define("MAJORITY", 18);

   echo MAJORITY;


?>
```

Instruction if

Les conditions en PHP sont définies grâce à des mots-clés tels que "if" (si), "else" (sinon) et "elseif" (sinon si).

```php
<?php
// On déclare une variable age avec comme valeur 16
$age = 16;

// On teste l'âge. Si l'âge est supérieur à 18, on affiche un message, sinon un autre message
if ($age >= 18) {
  echo 'adulte';
} else if ($age >= 13) {
  echo 'ado';
} else {
  echo 'enfant';
}
?>
```

Echo suite

L’instruction echo va nous permettre d’afficher un message mais aussi des variables avec guillet simple ou double.
La différence étant qu’avec guillemet double, php affichera le contenu de la variable alors que ce n’est pas le cas avec guillemet simple :

```php
<?php
$age = 30;

echo "age : $age ans"; // Affichage -> age : 30 ans
echo 'age : $age ans'; // Affichage -> age : $age ans

// Un raccourci pour echo intéressante quand on mélange avec du html :
?>
<p>Vous avez <?=$age; ?> ans</p>
```

Instruction switch

switch permet de comparer une expression à une liste de valeurs. Elle peut être utilisée de manière similaire à la structure "if"/"elseif"/"else", mais peut être plus concise et plus

```php
<?php
$numJour = 3;

switch ($numJour) {
    case 1:
        echo "Lundi";
        break;
    case 2:
        echo "Mardi";
        break;
    case 3:
        echo "Mercredi";
        break;
    default:
        echo "Numéro de jour invalide";
}
?>
```

php 8.0 a introduit l'instruction match

https://dev.to/mainick/php-match-expression-match-vs-switch-3j5b

Les opérateurs

- Opérateurs de comparaison de valeur :
  - ">" (strictement supérieur à)
  - "<" (strictement inférieur à)
  - ">=" (supérieur ou égal à)
  - "<=" (inférieur ou égal à)
  - "==" (égal à) ou "===" (pour vérifier le type)
  - "!=" (différent de) ou "!==" (pour vérifier le type)
- Opérateurs logiques en PHP :
  - "&&" (ET logique)
  - "||" (OU logique)
- Opérateurs arithmétiques :
  - "+" (addition)
  - "-" (soustraction)
  - "*" (multiplication)
  - "/" (division)
  - "%" (modulo)

Les tableaux

Un tableau (array en anglais) est une structure de données qui permet de stocker une liste d'éléments de manière organisée.

```php
<?php
    // Tableau simple
    $fruits = ["Banane", "Orange", "Pomme"];
    echo $fruits[0];  // On accède à un élément du tableau pas son index/clé, affiche
"Banane"
```

Note : Il existe un syntaxe alternative :

```php
$fruits = array("Banane", "Orange", "Pomme");
```

Les tableaux

Un tableau peut contenir des éléments de n'importe quel type (chaînes de caractères, nombres, objets, etc.). Chaque élément est composé d'une **clé/index** et de la **valeur**.

```php
<?php
    // Tableau associatif
    $utilisateur = ["nom" => "Dupond", "prenom" => "Jean", "age" => 26];
    echo $utilisateur["nom"];  // Affiche "Dupond"
?>
```

Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui contient d'autres tableaux en tant qu'élément. Cela permet de créer des structures de données plus complexes pour stocker et manipuler des informations.

```php
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

?>
```

Boucle foreach

La boucle "foreach" permet de parcourir un tableau et d'exécuter un bloc de code pour chaque élément du tableau. La boucle "foreach" prend en compte les **clés** et les **valeurs** du tableau.

Exemple sans clé

```php
<?php
    $fruits = ["Banane", "Orange", "Pomme"];

    foreach ($fruits as $key=>$value) {
        echo "<p>$value</p>";
    }
?>
```

Boucle foreach

Exemple avec clé

```php
<?php
    $fruits = ["Banane" => "Jaune", "Orange" => "Orange", "Pomme" => "Verte"];

    foreach ($fruits as $fruit => $couleur) {
        echo "<p>$fruit est de couleur $couleur</p>";
    }
?>
```

Boucle foreach

Exemple avec tableau multidimensionnel

```php
<?php
$utilisateurs = [
    ["nom" => "Dupond", "prenom" => "Jean", "age" => 26],
    ["nom" => "Martin", "prenom" => "Rose", "age" => 35],
    ["nom" => "Doe", "prenom" => "Jane", "age" => 31]
];

foreach ($utilisateurs as $key=>$utilisateur) {
    echo "<h2>Nom : " . $utilisateur["nom"] . "</h2>";
    echo "<p>Prénom : " . $utilisateur["prenom"] . "</p>";
    echo "<p>Age : " . $utilisateur["age"] . "</p>";
}
?>
```

Boucle for

La boucle "for" permet d'exécuter un bloc de code plusieurs.

La boucle "for" se compose de trois parties : l'initialisation, la condition de fin de boucle et l'incrémentation.

```php
<?php
    for ($i = 1; $i <= 10; $i++) {
        echo "$i ";
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Boucle while

La boucle "while" permet d'exécuter un bloc de code de manière itérative tant qu'une condition est vraie. Elle se compose d'une condition qui est évaluée avant chaque itération de la boucle.

Exemple sans clé

```php
<?php
    $i = 1;
    while ($i <= 10) {
        echo "$i ";
        $i++;
    }
    // Affiche : 1 2 3 4 5 6 7 8 9 10
?>
```

Les fonctions

Une fonction est un bloc de code qui peut être exécuté de manière autonome et qui peut éventuellement retourner une valeur. Une fonction peut être appelée à plusieurs reprises dans un programme, ce qui permet de réutiliser du code et de rendre celui-ci plus lisible et maintenable.

```php
<?php
function carre(float $nombre):float
{
    return $nombre * $nombre;
}

echo carre(5);  // Affiche 25

?>
```

On peut typer les arguments et le retour.
Même si ce n'est pas obligatoire, c'est fortement conseillé pour rendre son code plus robuste.

include vs require ?

Include et require

Les instructions "include" et "require" permettent d'inclure et d'exécuter un fichier au sein d'un autre fichier. Elles permettent de partager du code commun entre plusieurs fichiers et de rendre celui-ci plus lisible et maintenable.

```php
<?php
include 'fichier1.php';
require 'fichier2.php';
?>
```

La différence entre ces deux instructions est que require va générer une exception si le fichier est introuvable alors qu’include affichera seulement un warning.

Include_once et require_once

"include_once" et "require_once" fonctionnent de manière similaire à "include" et "require", mais avec une différence importante : si le fichier inclus a déjà été inclus dans le script en cours d'exécution, il ne sera pas inclus de nouveau.

```php
<?php
include_once 'fichier1.php';
require_once 'fichier2.php';
?>
```

LES FORMULAIRES

$_GET

$_GET est un tableau associatif qui contient les données envoyées au serveur via la méthode "GET" d'un formulaire HTML ou d'une URL.

Les données passées par une adresse url sont de la forme suivante :

mapage.php**?**nom**=**dupond**&**prenom**=**jean

On pourra ensuite y accéder en php de la manière suivante :

```php
<?php
echo $_GET['prenom'];
if (isset($_GET['nom'])) {
    echo $_GET['nom'];
}
?>
```

$_POST

$_POST est un tableau associatif qui contient les données envoyées au serveur via la méthode "POST" d'un formulaire HTML.

On pourra ensuite y accéder en php de la manière suivante:

```php
<?php
echo $_POST['prenom'];
if (isset($_POST['nom'])) {
    echo $_POST['nom'];
}
?>
```

Prévenir les failles XSS

Utiliser htmlspecialchars() avant d'afficher des données

```php
<?php
$dataFromDatabase = "<script>alert('XSS attack!');</script>";
echo htmlspecialchars($dataFromDatabase);
```

Si on souhaite autoriser certaines balises, on peut utiliser cette librairie:
http://htmlpurifier.org/

LES SESSIONS

Les sessions

Une session est une manière de stocker des données côté serveur pour un utilisateur spécifique pendant sa visite sur un site web. Elle permet de conserver des informations entre différentes requêtes HTTP.
En php on devra tout d'abord démarrer une session avec session_start().
On pourra ensuite manipuler les données de session avec le tableau $_SESSION.

```php
// Démarre une nouvelle session
session_start();

// Stocke des informations dans la session
$_SESSION['username'] = 'johndoe';
```

Les sessions : connexion

- Connecter un utilisateur passe par deux phases :
  - On l'authentifie avec son login et mot de passe
  - On persiste (on stocke) les informations dans une session

```php
session_start();

if ($_POST['username'] === "admin" && $_POST['password'] === "XD2Ka@" ) {
    //Prévient les attaques de fixation de session
    session_regenerate_id(true);
    $_SESSION['username'] = $_POST['username'];
}
```

Les sessions : logout

Lorsque l'on souhaite déconnecter un utilisateur, nous allons devoir supprimer les données de session avec session_destroy. L'appel à cette fonction va détruire les donnée du serveur mais pas vider le tableau $_SESSION.
Nous allons donc devoir supprimer le tableau de session avec un unset().
Nous appelons également session_regenerate_id(true); pour prévenir la fixation de session

```php
session_start();
//Prévient les attaques de fixation de session
session_regenerate_id(true);
//Supprime les données de session du serveur
session_destroy();
//Supprime les données du tableau $_SESSION
unset($_SESSION);
```

Rénérer l’ID de session

Régénérer l'ID de session après la connexion et la déconnexion : Après une connexion réussie ou une déconnexion, il est recommandé de régénérer l'ID de session en utilisant la fonction session_regenerate_id. Cela empêche les attaques de fixation de session, où un attaquant peut fixer l'ID de session avant que l'utilisateur ne se connecte.

```php
<?php
session_regenerate_id(true);
```

http://www.web-d.be/post/94/l'attaque-par-fixation-de-session,-et-comment-s'en-prot%C3%A9ger.html

Protéger le cookie de session

secure (ne fonctionnera pas en local sans https)
paramètre des cookies qui indique au navigateur que le cookie doit être envoyé uniquement via une connexion HTTPS sécurisée. Cela signifie que le cookie ne sera pas transmis sur une connexion HTTP non sécurisée.

httponly
paramètre des cookies qui restreint l'accès aux cookies via JavaScript. Lorsque l'attribut "HttpOnly" est défini sur true, le cookie ne peut être accédé ou modifié que par le serveur via des requêtes HTTP.

```php
<?php
session_set_cookie_params([
    'lifetime' => 3600,
    'path' => '/',
    'domain' => 'example.com',
    'secure' => true,
    'httponly' => true
]);
session_start();
```

Protéger le cookie de session

Secure
Paramètre des cookies qui indique au navigateur que le cookie doit être envoyé uniquement via une connexion HTTPS sécurisée. Cela signifie que le cookie ne sera pas transmis sur une connexion HTTP non sécurisée.
https://php.net/session.cookie-secure

SameSite
Paramètre des cookies qui contrôle le comportement de transmission des cookies en fonction de l'origine de la requête. Il permet de limiter les risques de certaines attaques, telles que les attaques de type Cross-Site Request Forgery (CSRF).
https://tools.ietf.org/html/draft-west-first-party-cookies-07

StrictSession
Lorsque le mode strict des sessions est activé, les cookies de session sont associés uniquement à l'adresse IP du client qui les a créés. Cela signifie que si l'adresse IP du client change entre les requêtes, la session sera invalidée.
https://wiki.php.net/rfc/strict_sessions

UseCookie et **UseOnlyCookie**
Permet de forcer l'utilisation de cookie pour la session (au lieu de le passer en get ou post)
https://php.net/session.use-cookies

ACCÈS AUX DONNÉES AVEC PDO

PDO

PDO (PHP Data Objects) est une extension de PHP. Elle permet de gérer les bases de données de manière uniforme, quel que soit le type de base de données utilisé (MySQL, SQLite, etc.). Avec PDO nous allons pouvoir nous connecter à une base de données, exécuter des requêtes SQL et récupérer les résultats. PDO offre également des fonctionnalités de sécurité pour protéger contre les injections SQL.

PDO

Ci-dessous un exemple d'une requête **NON SÉCURISÉ**

```php
<?php
$pdo = new PDO('mysql:dbname=my_db;host=localhost;charset=utf8mb4', 'root', '');
$id = $_GET['id'];
$query = $pdo->query("SELECT * FROM user WHERE id = $id");
$result = $query->fetch(PDO::FETCH_ASSOC);
?>
```

Ce code n'est pas sécurisé car un attaquant pourra injecter du code dans le paramètre d'url :

?id=5;DELETE FROM user;

La requête suivante sera alors exécutée :

SELECT * FROM user WHERE id =5;DELETE FROM user;

PDO

PDO nous permet de préparer, exécuter et récupérer des données.

On commence par instancier un objet PDO dans un try/catch pour gérer les erreurs.

```php
<?php
try
{
    $pdo = new PDO("mysql:dbname=my_db;host=localhost;charset=utf8mb4", "root", "");
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}
?>
```

PDO

Avec l’objet PDO nous pouvons effectuer des requêtes préparées

La préparation de requête avec bindParam() nous permet de sécuriser nos requêtes.

```php
<?php
$id = (int)$_GET['id'];
$query = $pdo->prepare("SELECT * FROM users WHERE id = :id");
$query->bindValue(':id', $id, PDO::PARAM_INT);
$query->execute();
//fetch() nous permet de récupérer une seule ligne
$result = $query->fetch(PDO::FETCH_ASSOC);
//$result est un tableau association qu’on peut manipuler comme on l’a vu précédemment
?>
```

PDO

Pour récupérer plusieurs lignes, on utilise fetchAll

```php
<?php
$query = $pdo->prepare("SELECT * FROM users");
$query->execute();
$result = $query->fetchAll(PDO::FETCH_ASSOC);
//$result sera un tableau contenant une ligne par utilisateur. Il faudra donc faire une foreach pour parcourir ce tableau
?>
```