PHP-MySQL*

Partie 4

R3.01 : Développement web

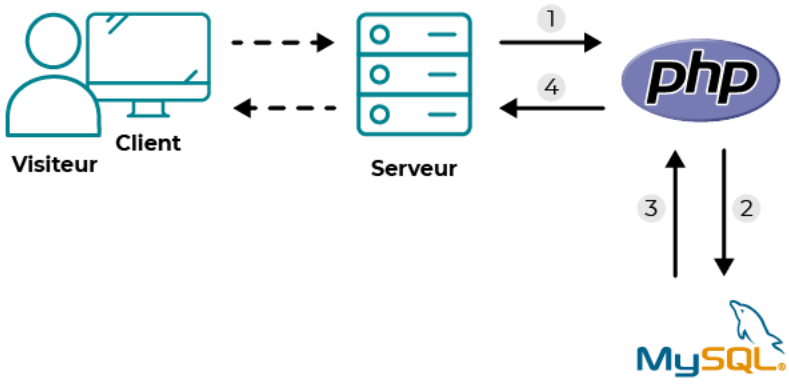
Objectif

Le principal objectif de cette partie du cours sera d'apprendre les instructions nécessaires à écrire en PHP pour effectuer des requêtes en base de données, et les bases du langage SQL.

1 Préliminaires

Comprenez comment PHP fait la jonction entre vous et MySQL

Pour compliquer un petit peu l'affaire (sinon, ce n'est pas rigolo), on ne va pas pouvoir parler à MySQL directement. Eh non, seul PHP peut le faire !
C'est donc PHP qui va faire l'intermédiaire entre vous et MySQL. On devra demander à PHP : "Va dire à MySQL de faire ceci".



Voici ce qui peut se passer lorsque le serveur a reçu une demande d'un client qui veut poster un message :

1. Le serveur utilise toujours PHP, il lui fait donc passer le message.

*https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql

2. PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL. En effet, le code PHP contient à un endroit "Va demander à MySQL d'enregistrer ce message". Il fait donc passer le travail à MySQL.
3. MySQL fait le travail que PHP lui a soumis et lui répond "OK, c'est bon !".
4. PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé.

Importez la base de données du projet fil rouge

Travail à faire

En utilisant phpPyAdmin ou la commande SQL `SOURCE`, chargez le fichier `creation_base.sql` disponible dans ce devoir.

2 Accédez aux données en PHP avec PDO

Vous allez maintenant communiquer avec votre base de données de recettes via PHP.

À la fin de cette section, vous aurez obtenu une nouvelle version de la page d'accueil du site, cette fois créée à l'aide de PHP et de MySQL.
Pour pouvoir travailler avec la base de données en PHP, il faut d'abord s'y connecter.

Il va donc falloir que PHP s'authentifie : on dit qu'il établit une connexion avec MySQL.

Une fois que la connexion sera établie, vous pourrez faire toutes les opérations que vous voudrez sur votre base de données.

Pour se connecter à une base de données MySQL, vous allez devoir utiliser une extension PHP appelée PDO ("PHP Data Objects"). Cette extension est fournie avec PHP.

2.1 Connectez PHP à MySQL avec PDO

Voici l'instruction PDO pour vous connecter à votre base `my_recipes` :

```php
<?php
$db = new PDO(
    'mysql:host=localhost;dbname=my_recipes;charset=utf8',
    'root',
    'root'
);
?>
```

Cette ligne de code crée donc une connexion à la base de données en indiquant dans l'ordre dans les paramètres :

- le nom d'hôte : `localhost` ;
- la base de données : `my_recipes` ;
- l'identifiant (login) : `root` ;
- le mot de passe (rappel : sous UwAmp, le mot de passe est `root`).

Testez la présence d'erreurs

Si vous avez renseigné les bonnes informations (nom de l'hôte, de la base, login et mot de passe), rien ne devrait s'afficher à l'écran.

Toutefois, s'il y a une erreur (vous vous êtes trompé de mot de passe ou de nom de base de données, par exemple), PHP risque d'afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe !

Vous ne voudrez pas que vos visiteurs puissent voir le mot de passe si une erreur survient lorsque votre site est en ligne. Il est préférable de traiter l'erreur.

En cas d'erreur, PDO renvoie ce qu'on appelle une exception, qui permet de « capturer » l'erreur :

```php
<?php
try
{
  $db = new PDO(
    'mysql:host=localhost;dbname=my_recipes;charset=utf8',
    'root',
    'root');
}
catch (Exception $e)
{
  die('Erreur : ' . $e->getMessage());
}
?>
```

Travail à faire

Dans votre dossier de travail, créez un fichier `mysql.php` et copiez-y le code ci-après :

```php
<?php

const MYSQL_HOST = 'localhost';
const MYSQL_PORT = 3306;
const MYSQL_NAME = 'my_recipes';
const MYSQL_USER = 'root';
const MYSQL_PASSWORD = 'root';

try {
    $db = new PDO(
        sprintf('mysql:host=%s;dbname=%s;port=%s;charset=utf8',
        MYSQL_HOST, MYSQL_NAME, MYSQL_PORT),
        MYSQL_USER,
        MYSQL_PASSWORD
    );
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(Exception $exception) {
    die('Erreur : '.$exception->getMessage());
}
?>
```

Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Connectez PHP à MySQL avec PDO"
> git push -u origin main
```

2.2 Effectuez une première requête SQL

L'objectif ici consiste à récupérer la liste des recettes qui étaient au départ dans une variable sous forme de tableau associatif, et qui seront dorénavant stockées dans votre base de données.
Effectuons la requête à l'aide de l'objet PDO :

```php
<?php
$recipesStatement = $db->prepare('SELECT * FROM recipes');
?>
```

Le problème, c'est que `$recipesStatement` contient quelque chose d'inexploitable directement : un objet de type PDOStatement. Cet objet va contenir la requête SQL que nous devons exécuter, et par la suite, les informations récupérées en base de données.
Pour récupérer les données, demandez à cet objet d'exécuter la requête SQL et de récupérer toutes les données dans un format "exploitable" pour vous, c'est-à-dire sous forme d'un tableau PHP :

```php
<?php
$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();
```

Une fois qu'on a récupéré les données sous forme d'un tableau PHP, on en revient à un tableau associatif que vous connaissez déjà.

Voici en résumé tout tout le code, de la connexion via PDO à l'affichage du résultat de la requête :

```php
<!-- On se connecte à MySQL -->
<?php include_once('mysql.php'); ?>

<!-- Si tout va bien, on peut continuer -->
<?php
// On récupère tout le contenu de la table recipes
$sqlQuery = 'SELECT * FROM recipes';
$recipesStatement = $db->prepare($sqlQuery);
$recipesStatement->execute();
$recipes = $recipesStatement->fetchAll();
?>
<!-- On affiche chaque recette une à une -->
<?php foreach ($recipes as $recipe) : ?>
  <p><?php echo $recipe['author']; ?></p>
<?php endforeach; ?>
```

Afficher les message d'erreurs

Lorsqu'une requête SQL « plante », bien souvent PHP vous dira qu'il y a eu une erreur à la ligne du `fetchAll` :

```
Fatal error: Call to a member function fetchAll() on a non-object in index.php on line 13
```

Ce n'est pas la ligne du `fetchAll` qui est en cause : c'est souvent vous qui avez mal écrit votre requête SQL quelques lignes plus haut. Pour afficher des détails sur l'erreur, il faut activer les erreurs lors de la connexion à la base de données via PDO.

Il faut adapter la création de l'objet `$db` pour activer la gestion des erreurs :

```php
$db = new PDO(
    'mysql:host=localhost;dbname=my_recipes;charset=utf8',
    'root',
    'root',
    [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION],
);
```

Désormais, toutes vos requêtes SQL qui comportent des erreurs les afficheront avec un message beaucoup plus clair.

Travail à faire

Dans le script `mysql.php`, ajoutez le paramètre qui permet d'activer les messages d'erreurs. Ensuite, modifiez votre requête `SELECT` du bloc de code précédent pour filtrer les recettes valides. Et intégrez ce bloc code dans votre projet de sorte que les recettes ne proviennent plus du fichier `variables.php` mais de la base de données.
Enfin, n'oubliez pas de retirer le tableau `$recipes` du fichier `variables.php`.

Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Effectuez une première requête SQL"
> git push -u origin main
```

2.3 Construisez des requêtes en fonction de variables

La requête que vous avez exécutée est simple et effectue toujours la même opération. Or, les choses deviennent intéressantes quand on utilise des variables de PHP dans les requêtes.

Identifiez vos variables à l'aide des marqueurs

Les marqueurs sont des identifiants reconnus par PDO pour être remplacés lors de la **préparation** de la requête par les variables PHP.

Le screencast « IdentifiezVariablesMarqueurs » disponible dans le dossier vous explique cette technique en détails.

En guise d'exemple complet, voici la requête pour retrouver les recettes valides de l'utilisateur Mathieu, écrite dans le respect des meilleures pratiques :

```php
<?php
$sqlQuery =
'SELECT * FROM recipes WHERE author = :author AND is_enabled = :is_enabled';

$recipesStatement = $db->prepare($sqlQuery);
$recipesStatement->execute([
'author' => 'mathieu.nebra@exemple.com',
'is_enabled' => true,
]);
$recipes = $recipesStatement->fetchAll();
]);
```

On ne concatène JAMAIS une requête SQL pour passer des variables, au risque de créer des injections SQL ! Le sujet des injections SQL sera abordé dans un autre cours. Si vous souhaitez en apprendre plus à ce sujet, je vous invite à consulter Wikipedia.

3 Ajoutez, modifiez et supprimez des recettes

Si vous voulez que des utilisateurs contribuent à votre site, il faudra leur proposer un formulaire de création et de modification.

Affichez les erreurs SQL

Comme vous le savez, le langage SQL est un langage à part entière dont on se sert en PHP. S'il peut y avoir des erreurs en PHP, il peut aussi y avoir des erreurs en SQL.
Il se peut par exemple que votre requête soit mal écrite, que la table que vous voulez ouvrir n'existe pas, etc. Bref, les erreurs possibles sont là encore nombreuses.

Repérez la requête qui selon vous plante, et forcez l'affichage de l'erreur s'il y en a une, comme ceci :

```php
<?php
$sqlQuery =
'SELECT * FROM recipes WHERE author = :author AND is_enabled = :is_enabled';
$recipesStatement = $db->prepare($sqlQuery);
$recipesStatement->execute([
'author' => 'mathieu.nebra@exemple.com',
'is_enabled' => true,
])
or die(print_r($db->errorInfo()));
```

Si la requête fonctionne, aucune erreur ne sera affichée. Si en revanche la requête plante, PHP arrêtera de générer la page et vous affichera l'erreur donnée par MySQL...

À partir de là, il va falloir vous débrouiller tout seul, car les erreurs SQL sont assez nombreuses et je ne peux pas toutes les lister.

En général, MySQL vous dit : « You have an error in your SQL syntax near 'XXX' ». À vous de bien relire votre requête SQL ; l'erreur se trouve généralement près de l'endroit où on vous l'indique.

3.1 Ajoutez une recette

Vous allez ajouter un formulaire de contribution de recettes.

Pour cela, vous aurez besoin de trois choses :

1. Ajouter un formulaire PHP de création de recettes.
2. Vérifier les données soumises en PHP.
3. À l'aide de PDO, exécuter l'insertion de la nouvelle recette en base de données.

Vous connaissez déjà les deux premières étapes, donc nous ne les détaillerons pas dans cette section, mais vous devrez produire le code complet dans le travail à faire suivant.

Ajoutez une recette avec l'instruction INSERT INTO

Voici par exemple une requête au sein d'un script PHP qui ajoute une recette :

```php
<?php
try
{
        $db = new PDO(
                'mysql:host=localhost;dbname=my_recipes;charset=utf8',
                'root', 'root');
}
catch (Exception $e)
{
        die('Erreur : ' . $e->getMessage());
}

// Ecriture de la requête
$sqlQuery = 'INSERT INTO recipes(title, recipe, author, is_enabled)
VALUES (:title, :recipe, :author, :is_enabled)';

// Préparation
$insertRecipe = $db->prepare($sqlQuery);

// Exécution ! La recette est maintenant en base de données
$insertRecipe->execute([
    'title' => 'Cassoulet',
    'recipe' => 'Etape 1 : Des flageolets ! Etape 2 : Euh ...',
    'author' => 'contributeur@exemple.com',
    'is_enabled' => 1, // 1 = true, 0 = false
]);
?>
```

Remarques

- Aucune valeur n'est passée pour le champ `recipe_id` car il a la propriété `AUTO_INCREMENT`, MySQL lui attribuera une valeur automatiquement.
- Une valeur entière est passée pour le champ `is_enabled` parce que cette information est stockée sous forme d'entier (0 pour FAUX et 1 pour VRAI).

- Vous récupèrerez des variables de `$_POST` (issues du formulaire) pour insérer une entrée dans la base de données.

Travail à faire

Le screencast « AjoutezRecette » disponible dans le dossier vous guide pas à pas pour :

1. ajouter un formulaire PHP de création de recettes,
2. puis vérifier les données soumises en PHP,
3. et enfin,à l'aide de PDO, exécutez l'insertion de la nouvelle recette en base de données.

Ajoutez un formulaire PHP de création de recettes dont l'accès est réserbé à un utilisateur connecté. Puis vérifiez les données soumises en PHP et enfin, à l'aide de PDO, exécutez l'insertion de la nouvelle recette en base de données.

Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Ajoutez une recette"
> git push -u origin main
```

3.2 Éditez une recette

Travail à faire

Vos utilisateurs souhaitent maintenant pouvoir modifier leurs recettes.

Le screencast « ModifiezRecette » disponible dans le dossier vous guide pas à pas pour réaliser cette nouvelle fonctionnalité d'édition de recette.

Ajoutez un formulaire PHP de d'édition de recette dont l'accès est réserbé à un utilisateur connecté et propriétaire de la recette en question. Puis vérifiez les données soumises en PHP et enfin, à l'aide de PDO, exécutez la modification de la recette en base de données.

Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Modifiez une recette"
> git push -u origin main
```

3.3 Supprimez une recette

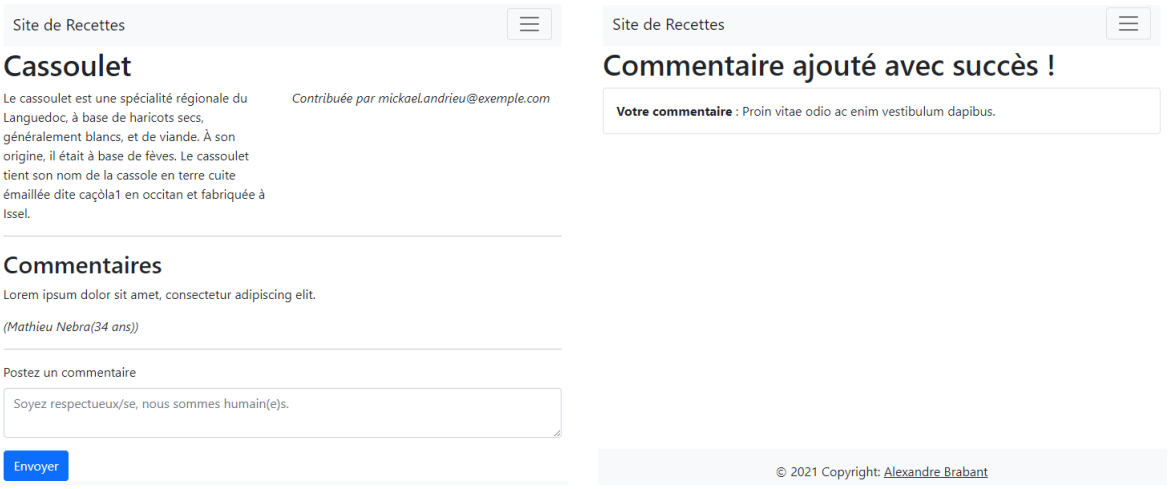
Devinez quoi ? Vos utilisateurs souhaitent pouvoir supprimer une recette...
Je vous laisse programmer par vous-même cette nouvelle fonctionnalité.

Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Supprimez une recette"
> git push -u origin main
```

4 Ajoutez un commentaire

Pour chaque recette affichée, permettre à l'utilisateur d'ajouter un commentaire.



Créez une nouvelle version de votre projet avec l'enchaînement de commandes ci-après :

```
> git add --all
> git commit -m "Ajoutez un commentaire"
> git push -u origin main
```

5 Pour aller plus loin

1. Affichez les commentaires du plus récent au plus ancien.

2. Dans le formulaire d'édition d'une recette, ajoutez une case à cochée pour la validation.

3. Modifiez la structure de la table `recipes` :
   (a) Ajoutez une colonne `user_id` (clé étrangère),
   (b) Supprimez la colonne `author`.
   ☞ Il y aura des modifications conséquentes à faire dans le code PHP…

4.
   (a) Ajoutez deux colonnes dans la table `comments` :
   `created_at` : date de publication du commentaire ;
   `review` : une note comprise entre 0 et 5 ;
   (b) Modifiez en conséquence le formulaire de saisie d'un commentaire.
   (c) Calculez et affichez la moyenne des notes obtenues pour une recette.