Accès aux données avec PHP

1 – Dialogue avec la base de données

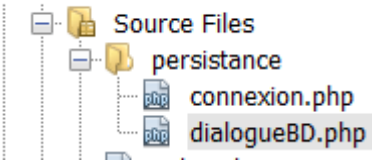
Contexte : Dans un premier temps, notre objectif est **d'extraire** et **d'afficher** la **liste des employés** de notre BD.

➔ Toujours dans le dossier "**persistance**", vous créez un nouveau fichier PHP "**dialogueBD.php**" qui contient une **classe nommée DialogueBD** dont le rôle sera **l'exécution des requêtes SQL**.

Elle possèdera les méthodes suivantes que nous programmerons :

- Lecture pour retrouver les données correspondant à un **select**
- Modification pour les requêtes de type **update**
- Ajout pour les requêtes de type **insert**
- Suppression pour les requêtes de type **delete**

Dans un premier temps, cette classe ne comportera qu'une méthode de lecture : **lecture de tous les employés** :



*Fichier **dialogueBD.php***

```php
<?php
require_once 'connexion.php';

class DialogueBD {
    public function getTousLesEmployes() {

        try {
            $conn = Connexion::getConnexion();
            $sql = "SELECT NomEmpl, PrenomEmpl FROM employe ORDER BY Matricule";
            $sth = $conn->prepare($sql);
            $sth->execute();
            $tabEmployes = $sth->fetchAll(PDO::FETCH_ASSOC);
            return $tabEmployes;
        } catch (PDOException $e) {
            $erreur = $e->getMessage();
        }
    }
}
```

Attention de bien respecter les **majuscules et minuscules**

dialogueBD.php pour le fichier
DialogueBD pour la classe

Explications :

```
$conn = Connexion::getConnexion();
```

⇨ Connexion à la BD : appel de la méthode getConnexion() de la classe Connexion qui retourne un objet connexion qu'on récupère dans $conn. (On remarque les :: car c'est une méthode statique)

```
$sql = "SELECT NomEmpl, PrenomEmpl FROM employe ORDER BY matricule";
```

⇨ Ecriture de la requête SQL dans une variable nommée ici $sql (on peut la nommer comme on veut) Cette requête qui n'est ici qu'une variable, consiste à sélectionner tous les employés, triés sur leur matricule.

```
$sth = $conn->prepare($sql);
```

⇨ Préparation de la requête. On obtient l'objet $sth ("Statement Handle" : poignée de déclaration)

Cette instruction a pour objectif d'avertir le SGBD/R qu'il va y avoir une requête à exécuter et qu'il faut donc qu'il se prépare en "montant" les tables qui sont dans le FROM.

Cette préparation de la requête se fait grâce à la méthode **prepare** appliquée à l'objet **$conn**.

Le symbole -> est l'équivalent PHP du **symbole .** du langage C#.

L'instruction **$conn->prepare()** se lit donc : "J'appelle la méthode **prepare** de mon objet **$conn**", "et je lui passe en paramètre la variable contenant la requête".

Remarque : Nous verrons plus tard que cette étape de préparation sera surtout nécessaire lorsqu'on aura besoin d'utiliser des requêtes paramétrées, c'est-à-dire des requêtes où des valeurs de condition (where) pourront être variables.

```
$sth->execute();
```

⇨ Exécution de la requête préparée grâce à la méthode execute appliquée à l'objet $sth

L'instruction **$sth->execute()** se lit donc : "J'appelle la méthode **execute** de mon objet **$sth**".

Remarque : Pour le moment, **les données sont encore côté serveur**.

```
$tabEmployes = $sth->fetchAll(PDO::FETCH_ASSOC);
```

⇨ Récupération du tableau associatif "résultat de la requête" contenant toutes les lignes de cette requête.

Cette ligne récupère les enregistrements de la requête, grâce à la méthode **fetchAll()** appliquée à l'objet **$sth**.

Cette méthode renvoie les lignes du résultat de la requête, dans le tableau associatif **$tabEmployes**.

Ce tableau associatif sera utilisé dans la couche "Présentation" c'est-à-dire l'affichage de la page.

```
return $tabEmployes;
```

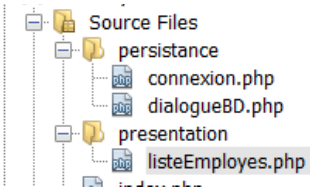
⇨ Retour du tableau (car getTousLesEmployes() est une fonction)

2 – Affichage des données

La connexion à la BD et les dialogues avec cette BD ont été créés dans la "**couche Persistance**".

L'affichage ou présentation des données va se faire dans la "**couche Présentation**".

➔ Vous ajoutez dans votre projet un nouveau dossier nommé **presentation** .



➔ Ajoutez une page PHP nommée **listeEmployes.php**.
(Cette page contiendra du code HTML et du code PHP).

Par la suite, le code PHP diminuera pour laisser la place à un langage de tags dans une architecture MVC (Model View Controller) utilisée par les frameworks (Laravel, Symfony …)

Cette page va afficher tous les employés : Elle va construire un objet référant la classe **DialogueBD** et appellera la méthode **getTousLesEmployes()** qui lui renverra un tableau associatif **$tabEmployes**

Code de la page ***listeEmployes.php***

```
<?php
// PARTIE DONNES ----------------------------------------------------------------------------------
// inclusion de la méthode de dialogue avec la BD
require_once '../persistance/dialogueBD.php';
try {
    // on créé un objet référant la classe DialogueBD
    $undlg = new DialogueBD();
    $mesEmployes = $undlg->getTousLesEmployes();
    }
catch (Exception $e) {
    $erreur = $e->getMessage();
} --                    ------------------------------------------------ --
?>
<!    PARTIE AFFICHAGE                                                    >
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Liste des employés</title>
    </head>
    <body>
        <?php
        if (isset($msgErreur)) {
            echo "Erreur : $msgErreur";
        }
        ?>
        <h1>Liste des employés</h1>
        <ul>
            <?php
            $nbemployes = 0;

            // Itération sur les lignes du tableau associatif (résultat requête
SQL)
            foreach ($mesEmployes as $ligne)
            {
                $nom = $ligne['NomEmpl'];
                $prenom = $ligne['PrenomEmpl'];
                echo "<li>$nom $prenom</li>";
                $nbemployes++;
            }
            ?>
```

➔ **Testez cette page**

Le résultat produit par cette page est bien sûr une page **H T M L** , renvoyée au navigateur.

(Faites un clic droit sur cette page, et observez le "code source de la page" … qui ne contient que du HTML !)

Reprenez votre code, et comprenez bien les explications qui suivent :

Explication du code :

```
require_once '../persistance/dialogueBD.php';
```

⇨ L' instruction require_once est similaire à include.

En cas d'échec, include ne produit qu'un avertissement alors que require stoppe le script.

```
$undlg = new DialogueBD();
$mesEmployes = $undlg->getTousLesEmployes();
```

⇨ Dans le bloc TRY, on crée un objet référant la classe DialogueBD : $undlg, puis on appelle la méthode (fonction) getTousLesEmployes sur cet objet, et on récupère le tableau associatif des lignes de la requête appelé ici $mesEmployes.

Dans la partie Affichage :

Le bloc PHP situé à l'intérieur de la balise HTML `< u l >`, permet de parcourir la liste des résultats de la requête SQL au moyen d'une boucle **foreach**.

A l'intérieur de la boucle, la variable **$ligne** permet d'accéder aux différents champs (colonnes) d'une ligne de résultat SQL. Il s'agit d'un tableau associatif dont **les clés sont les noms des champs** ou colonnes de la (ou les) table(s).

Ici, on utilise "NomEmpl" et "PrenomEmpl" pour accéder aux nom et prénom de chaque employé. On affiche ensuite ces informations dans un élément de liste (balise HTML `< l i >`).

Amélioration de l'application

➔ Modifiez la page **index.php** éventuellement créée avec le code suivant qui appelle la page **listeEmployes.php** par l'intermédiaire d'un lien <a href.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title> Menu principal </title>
    </head>
    <body>
        <h1> Menu principal Application PERSONNEL </h1>
        <br /><br />
        <a href="presentation/listeEmployes.php">Liste des employés</a><br />
    </body>
</html>
```

➔ Lancez votre application dans votre navigateur :

Menu principal Application PERSONNEL

Liste des employés

(Le fichier index.php est lancé par défaut)

Vous améliorerez le design plus tard,

A RETENIR :

L'architecture de ce projet est une première présentation du **principe du modèle MVC** :

- **Modèle**
  - Classes métiers avec les méthodes pour les traitements
  - Classes pour les données (accès et dialogue avec la BD)
- **Contrôleur**
  - Classe qui dialogue entre la vue et le modèle
- **Vue**
  - Couche de présentation : HTML + PHP qui affiche les données

Cette décomposition a pour but de bien séparer ce qui relève de **l'accès aux données** (géré uniquement en **P H P**) et **l'affichage de ces données** (géré avec **H T M L** et **P H P**).

.

Cette séparation données/affichage devra **toujours** être respectée.

L'objectif est de limiter le mélange entre PHP et HTML au strict nécessaire.

Si vous avez bien compris ce 1er script, vous pouvez passer à la suite (page suivante), sinon reprenez les explications de ce 1er script.

Application : Affichage des employés dans un tableau

On souhaite obtenir une page **tableauEmployes.php** qui affiche la liste des employés (matricule, nom, prénom) **dans un tableau** comme ci-dessous :

La démarche : Vous devez bien entendu vous inspirer des pages précédentes.

- Dans la couche "Persistance", modifiez simplement la fonction **getTousLesEmployes()** dans le fichier **dialogueBD.php**, en rajoutant le "**Matricule**" dans la requête.

Liste des employés

E001	DUBOIS	Roland
E002	GERNAU	Patricia
E003	LOUVEL	Marc
E004	MAUREL	Jeanne
E005	DUBOSC	Alain
E006	PARENT	Stéphanie
E007	POTIER	Jean
E008	FAUVEL	Anne
E009	NOUVION	Patrick
E010	BONNIER	Marie
E011	DURAND	Sylvie
E012	LENOIR	Carine

Total : 12 employés

- Rajoutez un **nouveau lien** "Tableau des employés" dans le menu de départ (index.php).