Bases de PHP

- Traiter une soumission de formulaire en PHP
- Mettre en place un architecture MVC avec PHP
- Développer un formulaire d'enregistrement en MVC

Traiter une soumission de formulaire en PHP

En PHP, la manière principale de communiquer des données au serveur est pas la soumission de formulaire. Voyons un exemple simple avec un formulaire de connexion.

Le formulaire

Commençons déjà par écrire notre formulaire dans un document HTML : `login.html`

Base du formulaire

```
<form method="post" action="login.php">


</form>
```

Nous allons utiliser la méthode HTTP POST car nous devons envoyer des données au serveur. Ensuite dans l'attribut `action` nous mettons le chemin sur le serveur denotre script PHP qui va gérer la soumission du formulaire.

Les champs

Ajoutons maintenant les champs dont nous avons besoin pour la connexion : un champs de type *text* pour le login et un champs de type *password* pour le mot de passe, ainsi qu'un boutton de soumission.

```
<form method="post" action="login.php">

  <input type="text" name="login" required/>
  <input type="password" name="password" required/>

  <button type="submit">Connexion</button>

</form>
```

L'attribut `required` permet d'empêcher la soumission du formulaire si le champs n'est pas remplit. Chaque champs possède une attribut `name` pour l'identifier lors du traitement dans le scipt PHP.

Voilà notre formulaire est prêt !

Le script PHP

Créez maintenant un script `login.php` afin de traiter la soumission du formulaire.

Récupération des données

Afin de récupérer des données envoyées via une requête POST, il faut faire appel dans le code PHP à la variable globale `$_POST`. `$_POST` (comme `$_GET`) est un dictionnaire clé valeur (ou tableau associatif) qui contient les valeurs des données passées dans le formulaire, avec pour clé, l'attribut `name` du champs dans le formulaire.

Mais avant de récupérer les valeurs, il faut vérifier qu'elles sont bien présentes grâce à la fonction `isset` :

```
if(!(isset($_POST["login"]) && isset($_POST["password"]))){
  Header("Location: index.html");
}
```

Si ce n'est pas le cas, on utilise `Header` pour rediriger vers le formulaire. On peut ensuite récupérer les valeurs des champs à partir de leurs noms :

```
$login = $_POST["login"];
$password = $POST["password"];
```

Après avoir récupéré ces données, on va pouvoir les valider, mais pour cela, il faut se connecter à une base de donnée !

Connexion à une base de donnée

Commencez par créer une table et des données sur votre base de donnée en exécutant le script suivant sur datagrip :

```
CREATE TABLE utilisateur (
  login VARCHAR(255),
  password VARCHAR(255)
);
INSERT INTO utilisateur(login,password) VALUES ("JohnTest","Test@2020");
```

Pour se connecter à une base de donnée MySQL avec PHP, il faut utiliser PDO :

```
$hostname = "localhost";
$base = "nom de la base de donnée";
```

```
$loginBD = "root";
$passBD = "mot de passe de la base de donnée";

$bdd = new PDO ( "mysql:server=$hostname; dbname=$base", "$loginBD", "$passBD", array
(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8'));
```

PDO (PHP Database Object) est orienté objet, c'est pour cela qu'on l'instancie avec le mot clé `new` et qu'on va devoir utiliser l'opérateur `->` dessus.

Pour éviter les injections SQL, il faut utiliser une requête préparée. Cela se fait avec la fonction `prepare` en utilisant des étiquettes dans la requête SQL :

```
$requete = $bdd->prepare ( "SELECT * FROM utilisateur WHERE login=:login AND
password=:password" );
```

On peut ensuite lier les paramètres avec la méthode `bindParams` :

```
$requete->bindParam ( ":login", $login );
$requete->bindParam ( ":password", $password );
```

Et enfin, on exécute la requête :

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

On peut ensuite récupérer la première ligne du résultat de la requête sous la forme d'un dictionnaire clé valeur (ou tableau associatif) avec la méthode `fetch()` :

```
$resultat = $resultatRequete->fetch();
```

Si la requête ne contient plus aucune ligne, on obtiens `false`. On peut donc vérifier que notre utilisateur est valide en faisant :

```
if($resultat){
  echo "Vous êtes connecté";
}
else {
  echo "Mauvais identifiants";
}
```

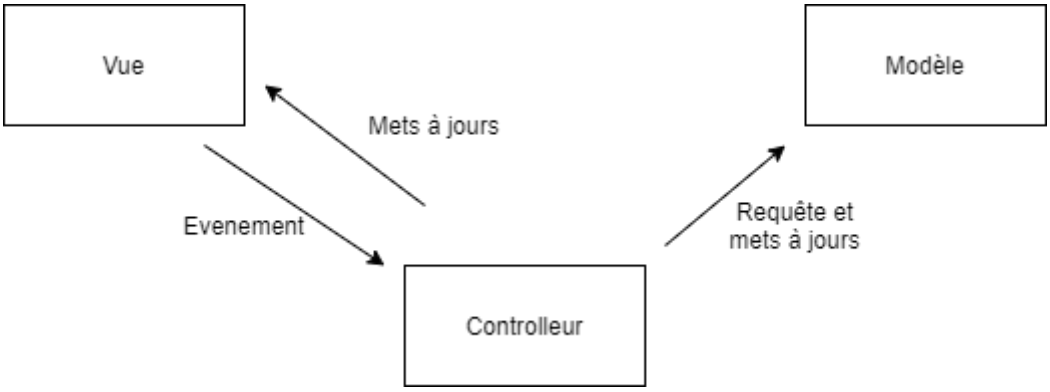
Voilà votre page de connexion est fonctionnelle, félifications □□

Mettre en place un architecture MVC avec PHP

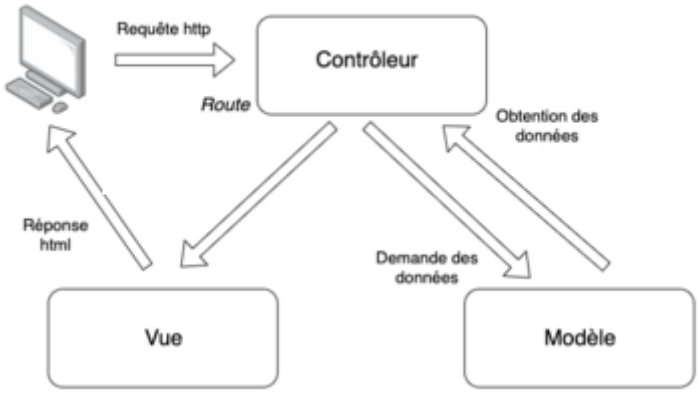
Le Modèle MVC

Le modèle MVC pour : Model - View - Controller est un Design Pattern visant à séparer une application en trois modules :

- Modèle : les données
- Controlleur : traite les données et mets à jours la vue
- Vue : Définit l'interface utilisateur



Si on applique au modèle du Web cela donne le schéma suivant :



MCV en PHP Vanilla

Commeçons par créer une arborescence de dossiers pour ranger nos scripts en fonction des différents modules de MVC :

- Un dossier `model` : contient des scripts qui contiennent des fonctions de requêtes SQL ainsi qu'un script de connexion à la base de donnée.
- Un Dossier `view` : va contenir des fichier `.tpl` qui à partir desquels seront générées des pages HTML.
- Un dossier `controller` : contient des scripts qui vont faire appel aux fonctions du `model` afin de récupèrer des données, les traiter, puis faire appel à la vue correspondante.

et un fichier `index.php` pour le routage.

Le routage

Pour accèder aux service des controleurs on va passer dans chaque requêtes une variable `controller` et une variable `action` en paramètre d'URL. Exemple : `http://localhost/index.html&controller=user&action=login`. Si bien qu'on fera toujours appel à `index.html`, qui lui s'occupera d'appeler la bonne action du bon controleur. Le code qui s'en occupe est le suivant :

```
if (isset ( $_GET ['controller'] ) & isset ( $_GET ['action'] )) {
$controle = $_GET ['controller'];
$action = $_GET ['action'];
} else { // absence de paramètres : prévoir des valeurs par défaut
$controle = "user";
$action = "login";
}

// Inclure le fichier php de contrôle
// Et lancer la fonction-action issue de ce fichier.

require ('./controller/' . $controle . '.php');
$action ();
```

Connexion à la base de donnée

On va créer un script de connexion à la base de donnée `connectDb` dans le dossier `model` et on y écrit la fonction suivante :

```
function connect(){
  $hostname = "localhost";
  $base = "nom de la base de donnée";
  $loginBD = "root";
  $passBD = "mot de passe de la base de donnée";

  try {
```

```
      $bdd = new PDO ( "mysql:server=$hostname; dbname=$base", "$loginBD", "$passBD", array
(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8'));
      return $bdd;
  }
  catch ( PDOException $e ) {
      echo "Echec de connexion à la base de donnée";
      die ( "Echec de connexion : " . $e->getMessage () . "\n" );
  }
}
```

Développer un formulaire d'enregistrement en MVC

Nous allons maintenant utiliser l'architecture MVC pour développer une fonctionalité simple d'enregistrement d'un utilisateur dans une base de donnée d'après un formulaire.

Vue

Commençons par créer notre vue, le formulaire, dans une fichier `register.tpl` dans le dossier `view` :

```
<form method="post" action="index.php?controller=registerController&action=register">

    <label for="lastname">Nom :
        <input type="text" name="lastname" required/>
    </label>
    <br>
    <label for="firstname">Prénom :
        <input type="text" name="firstname" required/>
    </label>
    <br>
    <label for="number">Numéro :
        <input type="text" name="number" required/>
    </label>
    <br>
    <label for="role">Rôle :
        <input type="text" name="role" required/>
    </label>
    <br>
    <label for="email">Email :
        <input type="text" name="email" required/>
    </label>

    <button type="submit">Inscription</button>
</form>
```

Modèle

On va maintenant créer un service de données : `registerService.php` dans le dossier `model`. Dans ce script, on va créer une fonction `registerUser` dans laquel on importe le script de connexion à la base de donnée pour accèder à l'objet PDO via notre fonction `connect()`. La fonction `registerUser` prend en paramètre les données correspondant au formulaire pour les insérer dans la base de donnée.

```
function registerUser($firstname, $lastname, $number, $role, $email)
{
    require("connectDb.php");
    $db = connect();
    $request = $db->prepare("INSERT INTO utilisateur(nom,prenom,num,role,email) value
(:firstname, :lastname, :number,:role, :email)");
    $request->bindParam(":firstname", $firstname);
    $request->bindParam(":lastname", $lastname);
    $request->bindParam(":number", $number);
    $request->bindParam(":role", $role);
    $request->bindParam(":email", $email);

    $request->execute();
    return $request;
}
```

Controleur

On va finalement créer un fichier `registerController.php` dans le dossier `controller`. Dans ce fichier, une fonction register va gérer l'affichage de l'interface ainsi que la soumission du formulaire.

```
function register()
{
    if (isset($_POST["lastname"]) && isset($_POST["firstname"]) && isset($_POST["number"]) &&
isset($_POST["role"]) && isset($_POST["email"])) {
        $lastname = $_POST["lastname"];
        $firstname = $_POST["firstname"];
        $number = $_POST["number"];
        $role = $_POST["role"];
        $email = $_POST["email"];
```

```
        require("./model/registerService.php");
        registerUser($firstname,$lastname, $number,$role,$email);
    }

    require("./view/register.tpl");
}
```

Si les données sont présentes, c'est que c'est une soumission du formulaire, sinon c'est simplement une requête de la page du formulaire, on doit donc faire une condition pour vérifier ça.

Voilà, vous avez un formulaire fonctionnel avec votre architecture MVC, félicitation □□