Développement WEB

Interactions Client-PHP-BD

Luc Dartois

luc.dartois@u-pec.fr

Objectifs

Objectifs du cours d'aujourd'hui :

- Passer des informations du client aux scripts PHP,
- Utiliser PHP pour lire/écrire sur une Base de Données,
- Faire la jonction client vers BD.

Problématique

La semaine dernière, on a vu comment utiliser PHP comme un langage "classique".
→ Cependant, nous voulons l'utiliser pour des applications Web.
→ Notamment pour permettre aux clients de transmettre des informations, comme par exemple :

- un login/mdp pour inscription,
- des fichiers à uploader,
- un post à publier,
- un panier à valider...

→ On a alors besoin de pouvoir relier les informations entrées par les clients jusqu'à la base de données.

Interaction : exemple de l'inscription

- Le client entre ses informations via un *formulaire*.

Interaction : exemple de l'inscription

- Le client entre ses informations via un *formulaire*.
- Ces infos sont envoyées à un script PHP, qui les traite.

Interaction : exemple de l'inscription

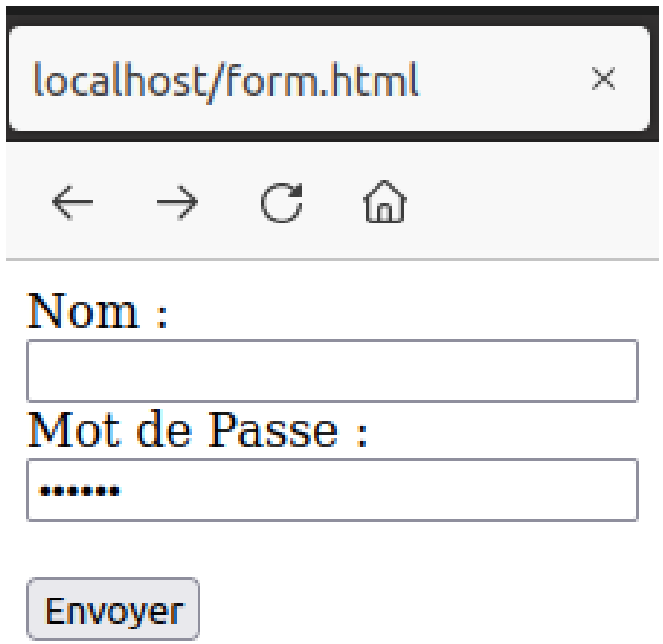
- Le client entre ses informations via un *formulaire*.
- Ces infos sont envoyées à un script PHP, qui les traite.
- Le script se connecte à la base de données.

Interaction : exemple de l'inscription

- Le client entre ses informations via un *formulaire*.
- Ces infos sont envoyées à un script PHP, qui les traite.
- Le script se connecte à la base de données.
- Il lance des requêtes de lecture/écriture à la base de données.

Rappels HTML : Formulaires

```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="mdp">Mot de Passe :</label><br>
  <input type="password" id="mdp" name="mdp" value="******">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```



Rappels HTML : Formulaires

```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="mdp">Mot de Passe :</label><br>
  <input type="password" id="mdp" name="mdp" value="******">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```

Un formulaire est :

- encadré dans une balise, `form` comportant :
  - une méthode, ici post
  - une action, indiquant l'url du script traitant les données du formulaire

Rappels HTML : Formulaires

```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="mdp">Mot de Passe :</label><br>
  <input type="password" id="mdp" name="mdp" value="******">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```

Un formulaire est :

- encadré dans une balise, `form`
- des questions, avec :
  - L'intitulé, dans une balise `label`, avec un pointeur "for",
  - Une zone de réponse, dans une balise `input`, avec un type, un id et un nom.

Rappels HTML : Formulaires

```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="mdp">Mot de Passe :</label><br>
  <input type="password" id="mdp" name="mdp" value="******">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```

Un formulaire est :

- encadré dans une balise, `form`
- des questions,
- un bouton d'envoi, dans une balise `input` de type submit, avec une value.

Passage au PHP : Méthode POST

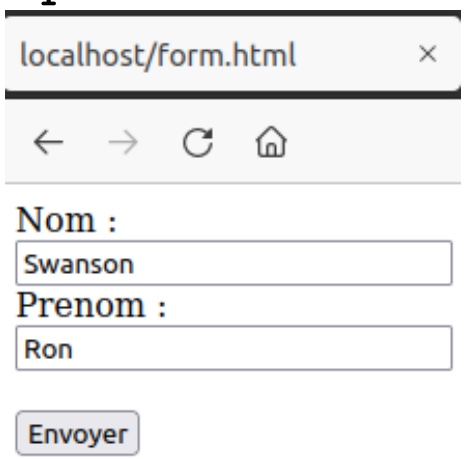
```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="prenom">Prenom :</label><br>
  <input type="text" id="mdp" name="prenom" value="">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```

Lorsque l'input de type "submit" est activé (par clic ou par entrée), le client est redirigé vers la page d'action (ici formAction.php), les informations étant stockées par la méthode indiquée (ici post).

Passage au PHP : Méthode POST

```
<form method="POST" action="formAction.php">
  <label for="nom">Nom :</label><br>
  <input type="text" id="nom" name="nom"><br>
  <label for="prenom">Prenom :</label><br>
  <input type="text" id="mdp" name="prenom" value="">
  <br><br>
  <input type="submit" value="Envoyer">
</form>
```

Le script récupérateur formAction.php possède un tableau `$_POST[]` contenant des couples (clé,valeur), où la clé correspond au *name* de la balise `input`.



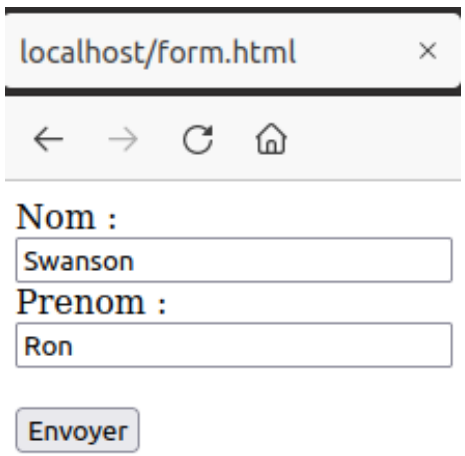
`$_POST[]` :

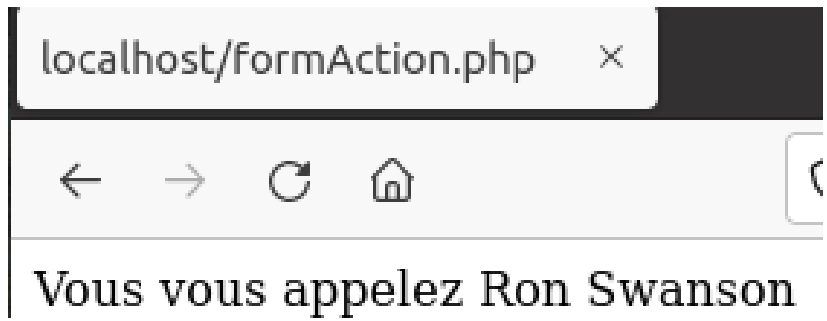
clé	valeur
nom	Swanson
prenom	Ron

Passage au PHP : Méthode POST (2)

formAction.php :

```
<html>
<body>
<?php
$prenom=$_POST['prenom'];          // On récupère les données
$nom=$_POST['nom'];
echo "Vous vous appelez ".$prenom." ".$nom; // On les traite
?>
</body>
</html>
```





Avant de continuer : Exemples d'input

Il existe de nombreux *types* pour la balise <input> :

Input de texte :

- email
- password
- hidden
- tel
- url
- ...

```
<label for="mdp"> Mot de passe :</label><br>
<input type="password" id="mdp" name="mdp">
```

Mot de passe :

Avant de continuer : Exemples d'input (2)

Le type *radio* permet un choix unique parmi une liste. On regroupe les boutons d'une même question grâce à l'attribut *name*. Le tableau `$_POST` contient le couple (name,value cochée).

```
Quel est le meilleur diagramme UML :<br>
<input type="radio" name="uml" value="dcu">
  <label for="dcu">DCU</label><br>
  <input type="radio" name="uml" value="dc">
  <label for="dc">DC</label><br>
  <input type="radio" name="uml" value="ds">
  <label for="ds">DS</label>
```

Quel est le meilleur diagramme UML :
- ○ DCU
- ◉ DC
- ○ DS

Avant de continuer : Exemples d'input (3)

Le type *checkbox* permet un choix multiple parmi une liste. On regroupe les boutons d'une même question grâce à l'attribut *name*. Le tableau $_POST contient un *tableau name* contenant en valeurs l'ensemble des cases cochées.

```
Quels diagrammes aimez-vous :<br>
 <input type="checkbox" name="CBuml" value="dcu">
  <label for="dcu">DCU</label><br>
  <input type="checkbox" name="CBuml" value="dc">
  <label for="dc">DC</label><br>
  <input type="checkbox" name="CBuml" value="ds">
  <label for="ds">DS</label>
```

Quels diagrammes aimez-vous :
- [x] DCU
- [x] DC
- [x] DS

Avant de continuer : Exemples d'input (3)

Le type *checkbox* permet un choix multiple parmi une liste. On regroupe les boutons d'une même question grâce à l'attribut *name*. Le tableau $_POST contient un *tableau name* contenant en valeurs l'ensemble des cases cochées.

```
$t=$_POST['CBuml'];
foreach($t as $v){
echo "Vous aimez $v !<br>";
}
```

Vous aimez dcu !
Vous aimez dc !
Vous aimez ds !

Passage au PHP : Méthode GET

```
<form method="GET" action="formAction.php">
//formulaire
  <input type="submit" value="Envoyer">
</form>
```

La méthode `GET` créé le tableau `$_GET[]`.
La différence étant que le contenu du tableau est inscrit dans l'url :

Passage au PHP : Méthode GET

```
<form method="GET" action="formAction.php">
//formulaire
  <input type="submit" value="Envoyer">
</form>
```

La méthode `GET` créé le tableau `$_GET[]`.
La différence étant que le contenu du tableau est inscrit dans l'url :

Il n'y a alors pas nécessairement besoin de passer par le formulaire, je peux créer mon propre tableau `$_GET[]` :

```
adresse_script?cle1=valeur1&cle2=valeur2 ...
```

GET vs POST

Les méthodes GET et POST jouent des rôles similaires, quand utiliser l'un ou l'autre ?

GET	POST
+Valeurs "bookmarkable"	− Valeurs temporaires
− Valeurs en clair (mdp)	+ Valeurs sécurisées par le http
− longueur limitée (2048 char)	+Pas de limite de longueur
+ méthode "idempotente"	

→ Cela dépend des cas !

Connecter un script à une Base de Données

Les données permanentes du serveur sont stockées sur une Base de Données. Il existe de nombreux systèmes de base de données.
Sur `dwarves.iut-fbleau.fr`, vous avez une base de données *MariaDB*, qui implémente le langage SQL (voir `https://mariadb.com/`).
→ Vous aurez besoin d'appliquer vos cours de SGBD !

Connecter un script à une Base de Données

Les données permanentes du serveur sont stockées sur une Base de Données. Il existe de nombreux systèmes de base de données.
Sur `dwarves.iut-fbleau.fr`, vous avez une base de données *MariaDB*, qui implémente le langage SQL (voir `https://mariadb.com/`).
→ Vous aurez besoin d'appliquer vos cours de SGBD !
On utilisera aussi phpMyAdmin, outils de gestion de BD avec php (voir `http://www.phpmyadmin.net`).

Rappels sur le modèle relationnel

Les données sont organisées en *tables*, qui ont un ensemble d'*attributs* (les colonnes).
Chaque attribut est typé.
On utilise SQL (Structured Query Language) pour effectuer des requêtes sur la base de données (lecture, écriture).

Rappels sur le modèle relationnel

Les données sont organisées en *tables*, qui ont un ensemble d'*attributs* (les colonnes).
Chaque attribut est typé.
On utilise SQL (Structured Query Language) pour effectuer des requêtes sur la base de données (lecture, écriture).

Ex :

Nom	Prénom	Email
Truc	muche	tm@gmail.com
Toto	tata	tata@toto.fr
Swanson	Ron	NULL

```
SELECT Nom From User where Email <> NULL;
```

Va afficher Truc, Toto.

Interaction PHP-BD

Les interactions entre scripts PHP et BD se déroulent de la façon suivante :

1. Connexion à la base de données
2. Envoi d'une requête
3. Exploitation des résultats
4. (Optionnel) Retour à (2)
5. Déconnexion.

Interaction PHP-BD

Les interactions entre scripts PHP et BD se déroulent de la façon suivante :

1. Connexion à la base de données
2. Envoi d'une requête
3. Exploitation des résultats
4. (Optionnel) Retour à (2)
5. Déconnexion.

→ Chaque script doit se connecter !
En pratique, on créé un fichier de connexion, qui sera inclut par chaque script devant dialoguer avec la BD.

Interaction PHP-BD

Les interactions entre scripts PHP et BD se déroulent de la façon suivante :

1. Connexion à la base de données
2. Envoi d'une requête
3. Exploitation des résultats
4. (Optionnel) Retour à (2)
5. Déconnexion.

→ Chaque script doit se connecter !
En pratique, on créé un fichier de connexion, qui sera inclut par chaque script devant dialoguer avec la BD.
Pour tout ça, on utilise une extension de PHP. Pour nous, ce sera *mysqli* (d'autres existent, comme *PDO_MYSQL*).

L'extension Mysqli

Mysqli est une API. Elle fournit tous les services permettant à PHP d'exécuter des requêtes SQL sur une BD.
Mysqli s'utilise globalement comme Mysql, avec un préfixe différent.
Vous trouverez la liste des méthodes de l'interface Mysqli à l'adresse : https://www.php.net/manual/fr/book.mysqli.php.

Un exemple

On possède une base de données *Chanson* comprenant diverses informations sur un ensemble de chansons. Elle est de la forme suivante :
`Chanson (idChanson, Titre, Artiste, Album, Genre, Année)`
$\rightarrow$ Le but est d'afficher un tableau HTML contenant les titres et artistes, et classé par genre.

Exemple : connexion

Première étape, se connecter à la base de données. On utilise `mysqli_connect` qui renvoie un objet représentant la connexion au server, ou false si cela échoue :

```
$db = mysqli_connect("hostname","user","password","Spotifieur"
if(!$db){
die("Connexion BD impossible");
}
```

Exemple : requête

On utilise ensuite `mysqli_query` pour effectuer la requête :

```
$resultat=mysqli_query($db,
"SELECT Titre, Artiste FROM Chanson ORDER BY Genre");
```

Exemple : requête

On utilise ensuite `mysqli_query` pour effectuer la requête :

```
$resultat=mysqli_query($db,
"SELECT Titre, Artiste FROM Chanson ORDER BY Genre");
```

La fonction `mysqli_query` renvoie un objet `mysqli_result` contenant le résultat de la requête en mémoire tampon.

Exemple : Exploitation

On utilise alors l'objet $resultat pour construire notre page HTML :

```
<table>
<tr> <th> Artiste </th> <th> Titre </th> </tr>
<?php
if($resultat){
while($chanson=mysqli_fetch_assoc($resultat)){
echo "<tr><td>{$chanson['Artiste']}</td>
<td>{$chanson['Titre']}</td><tr>";}}
else{
die("Erreur dans la requête");}
?> </table>
```

Exemple : Exploitation

On utilise alors l'objet $resultat pour construire notre page HTML :

```
<table>
<tr> <th> Artiste </th> <th> Titre </th> </tr>
<?php
if($resultat){
while($chanson=mysqli_fetch_assoc($resultat)){
echo "<tr><td>{$chanson['Artiste']}</td>
<td>{$chanson['Titre']}</td><tr>";}}
else{
die("Erreur dans la requête");}
?> </table>
```

La méthode `mysqli_fetch_assoc` renvoie à chaque appel, sous forme de tableau une ligne de la requête, ou l'objet `NULL` quand $resultat est vide.

Exemple : déconnexion

À la fin de notre script, il ne faut pas oublier de supprimer la connexion à la base de données :

```
mysqli_close($db);
```

Récapitulatif

Connexion

`$db=mysqli_connect("host","usr","pwd","db")` se connecte et renvoie un objet "connexion" $db.

Requête

`$res=mysqli_query($db,"Requête")` effectue la requête et renvoie un objet "résultat" $res.

Requête

`$l=mysqli_fetch_assoc($res)` renvoie une ligne du résultat sous forme tableau (attribut,valeur).

Déconnexion

`mysqli_close($db)` ferme la connexion ouverte au début (et détruit l'objet).

Requêtes préparées

Le système MySQL supporte les requêtes préparées. Une requête préparée ou requête paramétrable est utilisée pour exécuter la même requête plusieurs fois, avec une efficacité améliorée de part la préparation.
Cette préparation est gérée par le serveur de BD, cependant il faut alors :

- Préparer la requête : donner au serveur un squelette de requête qu'il doit s'attendre à recevoir
- Exécuter cette requête : une fois préparée, elle peut s'exécuter de nombreuses fois.

→ C'est par exemple utile lorsque l'on veut remplir une base de données.

Préparation

La fonction `mysqli_prepare` a la même syntaxe que `mysqli_query`: le premier argument est un objet mysqli (la connexion au serveur), et le 2e un String (la requête).

```
$db = mysqli_connect("hostname","user","password","Spotifieur"
if(!$db){
die("Connexion BD impossible");
}
$stmt = mysqli_prepare($db,
"INSERT INTO Artiste (Pseudo, Année)
VALUES (?,?)");
```

Les marqueurs ? sont les paramètres de la requête préparée.
`mysqli_prepare` renvoie un objet `mysqli_stmt`, qui va être utilisé pour l'exécution.

Exécution (1)

La fonction `mysqli_stmt_bind_param` permet de donner des valeurs aux paramètres de la requête. Elle prend en argument un `mysqli_stmt`, les valeurs à donner ainsi qu'un argument donnant leur type !

```
$pseudo="The Doors";
$annee=1965;
mysqli_stmt_bind_param($stmt,"si",$pseudo,$annee);
```

Le 2e argument sert à donner les types des valeurs :

- i pour un entier
- d pour un réel
- s pour un String
- b pour BLOB (Binary Large OBject)

Exécution (1)

La fonction `mysqli_stmt_bind_param` permet de donner des valeurs aux paramètres de la requête. Elle prend en argument un `mysqli_stmt`, les valeurs à donner ainsi qu'un argument donnant leur type !

```
$pseudo="The Doors";
$annee=1965;
mysqli_stmt_bind_param($stmt,"si",$pseudo,$annee);
```

Le 2e argument sert à donner les types des valeurs :

- i pour un entier
- d pour un réel
- s pour un String
- b pour BLOB (Binary Large OBject)

La requête peut ensuite être exécutée:

```
mysqli_execute($stmt);
```

Requête préparées, Infos complémentaires

Pour les requêtes d'écriture (UPDATE, DELETE, INSERT), il est possible de récupérer le nombre de lignes affectées avec :

```
$i=mysqli_stmt_affected_rows($stmt);
```

Requête préparées, Infos complémentaires

Pour les requêtes d'écriture (UPDATE, DELETE, INSERT), il est possible de récupérer le nombre de lignes affectées avec :

```
$i=mysqli_stmt_affected_rows($stmt);
```

Pour les requêtes de lecture (SELECT), on peut lier le résultat à des variables, et lire chaque ligne retournée :

```
$stmt=mysqli_prepare($db,
"SELECT Titre,Genre FROM Chanson");
mysqli_execute($stmt);
mysqli_stmt_bind_result($stmt,$t,$g);
while(mysqli_stmt_fecth($stmt)){
echo "$t est de style $g";
}
```

Requête préparées, Infos complémentaires

Pour les requêtes d'écriture (UPDATE, DELETE, INSERT), il est possible de récupérer le nombre de lignes affectées avec :

```
$i=mysqli_stmt_affected_rows($stmt);
```

Pour les requêtes de lecture (SELECT), on peut lier le résultat à des variables, et lire chaque ligne retournée :

```
$stmt=mysqli_prepare($db,
"SELECT Titre,Genre FROM Chanson");
mysqli_execute($stmt);
mysqli_stmt_bind_result($stmt,$t,$g);
while(mysqli_stmt_fecth($stmt)){
echo "$t est de style $g";
}
```

On peut aussi récupérer le résultat comme avant avec :

```
$res=mysqli_stmt_get_result($stmt);
```