PHP et MySQL avec PDO

Sommaire

WEBFORCE3
★★★<HELLO WORLD>

- Récupérer des données avec PDO.
- Les types de retours.
- fetch()
- fetchAll()
- fetchColumn()
- Les attaques XSS.
- Principe des injections SQL.
- Les Requêtes préparées.

PDO: Récupérer des données

Il existe 3 fonctions pour récupérer des données d'un SELECT:

- fetch() récupère la ligne suivante d'un jeu d'enregistrements:
  - $result = $sth->fetch(PDO::FETCH_ASSOC);
  - $result = $sth->fetch(PDO::FETCH_NUM);
  - $result = $sth->fetch(PDO::FETCH_BOTH);
  - $result = $sth->fetch(PDO::FETCH_OBJ);

- fetchAll() retourne un tableau contenant toutes les lignes du jeu d'enregistrements.

- fetchColumn() retourne une colonne depuis la ligne suivante d'un jeu de résultats.

PDO: fetch()

WEBFORCE3 ★★★<HELLO WORLD>

```php
<?php
include "connexionPDO.php";

$rq = "SELECT nomCommune from INSEE WHERE codePostal='02500'";
$stmt = $dbh->query($rq);
while($row = $stmt->fetch(PDO::FETCH_ASSOC))
{
  echo $row['nomCommune']."<br>";
}
$dbh = null;

?>
```

PDO: fetchAll()

WEBFORCE3 ★★★<HELLO WORLD>

```php
<?php
include "connexionPDO.php";


$rq = "SELECT * from INSEE WHERE codePostal='02500'";
$stmt = $dbh->query($rq);
$row = $stmt->fetchALL(PDO::FETCH_ASSOC);
echo "<pre>";
print_r($row);
echo "</pre>";
$dbh = null;
?>
```

PDO: fetchColumn()

WEBFORCE3
★★★<HELLO WORLD>

```php
<?php
include "connexionPDO.php";

$rq2 = "SELECT COUNT(nomCommune) FROM INSEE WHERE codePostal='02500'";
$stmt = $dbh->query($rq2);
$nbCommunes = $stmt->fetchColumn();
echo $nbCommunes."<br/>";

?>
```

Exercice pratique:

Reprendre le script de contrôle du login/password et adapter la méthode de récupération des infos.

Exercice pratique:

Utiliser la méthode FetchAll() et la fonction afficheTableau2D pour afficher des informations

Attaques XSS

WEBFORCE
★★★<HELLO WORLD>

- Le cross-site scripting (abrégé XSS), est un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, permettant ainsi de provoquer des actions sur les navigateurs web visitant la page.
- Les possibilités des XSS sont très larges puisque l'attaquant peut utiliser tous les langages pris en charge par le navigateur (JavaScript, Java, Flash...) et de nouvelles possibilités sont régulièrement découvertes notamment avec l'arrivée de nouvelles technologies comme HTML5.
- Il est par exemple possible de rediriger vers un autre site pour de l'hameçonnage ou encore de voler la session en récupérant les cookies.
- Le cross-site scripting est abrégé XSS pour ne pas être confondu avec le CSS (feuilles de style)1, X se lisant « cross » (croix) en anglais.
- Il existe deux types d'attaques XSS:
  - Non permanentes si les données saisies par l'utilisateur sont utilisées telles quelles par les scripts pour produire une page résultat.
  - Permanente si les données sont stockées sur le serveur (BDD, fichier, …). Ces attaques permettent d'atteindre un grand nombre de victimes avec une seule attaque (sur un forum par ex).

Attaques XSS
Démonstration

WEBFORCE3
★★★<HELLO WORLD>

```
<html>
    <form method="post" action="recherche.php">
        <input type="texte" name="recherche" />
        <input type="submit" value="Rechercher" />
    </form>
</html>

<?php
    echo "Bonjour ".$_POST['pseudo']." !"
?>
```

Attaques XSS Protection

- http://www.commentcamarche.net/contents/50-xss-cross-site-scripting
- https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web/la-faille-xss-1

Pour s'en protéger:

- htmlspecialchars() qui filtre les '<' et les '>'
- htmlentities(), idem htmlspecialchars() mais filtre tous les caractères équivalents au codage HTML ou JS.
- strip_tags() qui supprime toutes les balises.

Exercice pratique: Amusez vous avec un formulaire…

Injection SQL

Une injection SQL est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données, en injectant une requête SQL non prévue par le système et pouvant compromettre sa sécurité.

Par exemple, soit la requête:

```
SELECT uid FROM Users WHERE name = '(nom)' AND password = '(mot de passe hashé)';
```

```
SELECT uid FROM Users WHERE name = 'Dupont' AND password = '45723a2af3788c4ff17f8d1114760e62';
```

Si on envoie au script:

- Utilisateur: `Dupont';--` (en SQL -- est un début de commentaire).
- Mot de passe: peu importe.

La requête devient:

```
SELECT uid FROM Users WHERE name = 'Dupont'; -- ' AND password = '4e383a1918b432a9bb7702f086c56596e';
```

Injection SQL

https://fr.wikipedia.org/wiki/Injection_SQL

http://www.mcherifi.org/hacking/tutoriel-sql-injection-les-classiques.html

Pour s'en protéger:

- Filtrer les données fournies par l'utilisateur (intval(), floatval(), …).
- La fonction quote() de PDO place des guillemets simples autour d'une chaine d'entrée, si nécessaire et protège les caractères présents dans la chaine d'entrée, en utilisant le style de protection adapté au pilote courant.
- Utiliser des requêtes préparées.

Exercice pratique:

Tester le formulaire login/password et voir comment améliorer la sécurité

PDO: Requêtes Préparées

WEBFORCE3 ★★★<HELLO WORLD>

```
<?php

/* Exécute une requête préparée en passant un tableau de valeurs */
$sth = $dbh->prepare('SELECT nom, couleur, calories
    FROM fruit
    WHERE calories < ? AND couleur = ?');

// exécution d'une première requête
$sth->execute(array(150, 'rouge'));
$red = $sth->fetchAll();

Exécution d'une seconde requête.
$sth->execute(array(175, 'jaune'));
$yellow = $sth->fetchAll();


?>
```

Exercice pratique:

Reprendre le formulaire login/password et modifier les requêtes.

Exercices pratiques